Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Kristian Žarn

Gradnja 3D modelov predmetov iz barvnih slik z načrtovanjem najboljšega naslednjega pogleda

magistrsko delo

magistrski program druge stopnje
računalništvo in informatika

Mentor: izr. prof. dr. Danijel Skočaj

Ljubljana, 2019

Zahvala

Zahvaljujem se svojemu mentorju izr. prof. dr. Danijelu Skočaju za pomoč, družini za podporo in sestri Ajdi za pomoč pri rekonstrukciji.

Kristian Žarn, 2019

Kazalo

Seznam uporabljenih kratic

kratica	**angleško**	**slovensko**
SfM	Structure from Motion	Struktura iz gibanja
MVS	Multi-view Stereo	Večpogledni stereo
NBV	Next Best View	Najboljši naslednji pogled
SLAM	Simultaneous Localization and Mapping	Sočasna lokalizacija in kartiranje
SIFT	Scale Invariant Feature Transform	Velikostno invariantna transformacija značilnic
RANSAC	Random Sample Consensus	Konsenz naključnih vzorcev
GSD	Ground Sampling Distance	Razdalja vzorčenja terena
PPA	Pixels per Area	Število slikovnih elementov na površini
ICP	Iterative Closest Point	Iterativna metoda najbližje točke
SVD	Singular Value Decomposition	Razcep na singularne vrednosti
PCA	Principal Component Analysis	Analiza glavnih komponent
IP	Internet Protocol	Internetni protokol
URL	Uniform Resource Locator	Enotni naslov vira

Povzetek

Naslov: Gradnja 3D modelov predmetov iz barvnih slik z načrtovanjem najboljšega naslednjega pogleda

Rekonstrukcija geometrije iz barvnih slik je eden izmed klasičnih problemov računalniškega vida. Kvaliteta pridobljenega 3D modela je močno odvisna od zajetih slik. Ročno zajemanje je lahko dolgotrajno opravilo, pri katerem želimo s slikami doseči ustrezno natančnost in pokritost modela. Uporabnik, ki med zajemanjem nima nobene povratne informacije o ustreznosti slik, s težavo upošteva vse predpostavke algoritmov, kar je lahko vzrok za neuspešno rekonstrukcijo, ponovno zajemanje pa je lahko zelo drago ali celo nemogoče. V tem delu smo se osredotočili na razvoj postopka in programske opreme, ki podpira celoten proces rekonstrukcije. Za vsako zajeto sliko dobi uporabnik sprotno informacijo o njeni ustreznosti ter oceni kvalitete trenutnega 3D modela. Razvili smo tudi novo metodo za načrtovanje naslednjih pogledov, ki sistematično izboljšajo kvaliteto rekonstrukcije. Metoda temelji na novi meri za oceno kvalitete 3D modela, za katero pokažemo, da je njen koeficient linearne povezanosti z dejansko natančnostjo boljši od obstoječe mere. Pokažemo tudi, da je rekonstrukcija z našo metodo načrtovanja naslednjega pogleda primerljiva in v nekaterih primerih boljša od rekonstrukcije z enakomerno postavljenimi kamerami po navidezni polkrogli.

Ključne besede

računalniški vid, 3D rekonstrukcija, struktura iz gibanja, najboljši naslednji pogled

Abstract

Title: Building 3D models of objects from color images with next best view planning

Reconstruction of geometry from RGB images is one of the classic computer vision problems. The quality of the produced 3D model heavily depends on the input images. Manual image acquisition can be a lengthy process with which we want to attain the desired accuracy and completeness of the model. A user that has no feedback about the suitability of the images during the acquisition process can have difficulties complying with assumptions of the algorithms, which can result in an unsuccessful reconstruction. Additional image acquisition can be expensive or even impossible. In this work, we focus on development of a system and software that support the entire reconstruction process. The user gets online information about the adequacy of every captured image and an estimate of quality for the current 3D model. We also present a novel method for next best view planning, that systematically improves the quality of reconstruction. The method is based on a new quality measure. We show that the linear correlation coefficient between our measure and accuracy is better than that of the existing measure. We also show that the reconstruction obtained by the next best view planning is comparable and in some cases better than reconstruction with evenly spaced camera configuration in the shape of a hemisphere.

Keywords

computer vision, 3D reconstruction, structure from motion, next best view

Poglavje 1

Uvod

1.1 Motivacija

Rekonstrukcija 3D modelov iz slik je problem na področju računalniškega vida s katerim se raziskovalci ukvarjajo že več kot 30 let. Na podlagi zajetih slik želimo pridobiti 3D obliko, ki se predmetu najbolj prilega. Razviti postopki in algoritmi se uspešno uporabljajo v industriji, npr. za 3D kartiranje in navigacijo (Google maps, Apple maps), spletne trgovine, 3D tisk, posebne učinke v filmih, računalniške igre in arhiviranje kulturne dediščine. Sodobne digitalne kamere lahko zaradi visoke ločljivosti in kvalitete slik ob ustrezni uporabi proizvedejo 3D modele visoke kakovosti, ki je v določenih pogojih primerljiva z laserskim skeniranjem [1]. Njihova prednost v primerjavi z globinskimi kamerami ali laserskimi skenerji je cenovna dostopnost in enostavnost uporabe.

Kvaliteta rekonstrukcije, ki se običajno meri z natančnostjo in pokritostjo, je močno odvisna od zajetih slik. Natančnost se meri z razdaljo od rekonstrukcije do referenčnega modela, pokritost pa z razdaljo v obratni smeri. Natančna definicija je podana v poglavju 6.1.3. Ročno zajemanje je dolgotrajno opravilo, pri katerem želimo s slikami doseči ustrezno kvaliteto 3D modela. Algoritmi za rekonstrukcijo imajo nekatere predpostavke in omejitve, na primer: kot med pari sosednjih pogledov ne sme biti prevelik, translacija

pa ne premajhna, sosednji sliki se morata dovolj prekrivati, rekonstruirani predmeti morajo imeti dovolj dobro teksturo, ki pa ne sme biti preveč ponavljajoča, zaželeno je tudi, da je osvetlitev čimbolj enakomerna. Uporabnik, ki med zajemanjem nima nobene povratne informacije o ustreznosti slik, težko upošteva vse omenjene predpostavke. Lahko se zgodi, da zaradi neustreznih slik nekateri deli predmeta ne morejo biti rekonstruirani, ali pa ni dosežena želena natančnost modelov. Ponovno zajemanje slik je lahko zelo drago ali celo nemogoče, kot na primer pri dokumentiranju gradbišča, rekonstrukciji iz zračnih posnetkov in arhiviranju arheoloških najdb. Poleg tega se pogosto zajamejo tudi redundantne slike, ki povečajo čas procesiranja, brez pomembnega prispevka h kvaliteti modela.

Omenjene težave so pogoste v klasičnih pristopih, kjer je gradna 3D modela izvedena ločeno, po zajemanju slik. Z bolj zmogljivimi računalniki, boljšimi algoritmi in možnostjo izvajanja nekaterih algoritmov na grafičnih karticah so se pojavile tudi rešitve [2], ki gradijo grob 3D model že med zajemanjem slik in uporabniku dajejo sprotno informacijo o njegovi kvaliteti. Tvorba končnega podrobnega modela pa je še vedno izvedena ločeno. Na ta način lahko uporabnik optimizira proces zajemanja slik, odpre pa se tudi možnost za avtonomno rekonstrukcijo z roboti. Problem avtonomne rekonstrukcije naslavljajo algoritmi za načrtovanje najboljšega naslednjega pogleda (angl. next best view planning), ki skušajo najti takšno postavitev kamere, ki čimbolj izboljša trenutno kvaliteto rekonstrukcije.

V delu smo se osredotočili na razvoj postopka in programske opreme, ki podpira celoten proces rekonstrukcije, od zajemanja slik do tvorjenja končnega 3D modela s teksturo. Za vsako zajeto sliko dobi uporabnik sprotno informacijo o njeni ustreznosti in trenutni oceni kvalitete 3D modela. Na podlagi tega lahko uporabnik bolje presodi in načrtuje zajemanje nadaljnjih slik, poleg tega pa mu je predlagan tudi najboljši naslednji pogled. Naše delo je tudi korak v smeri avtonomne rekonstrukcije z roboti (npr. z robotskim manipulatorjem, kvadrokoperjem itd.).

1.2 Sorodna dela

Celoten postopek gradnje 3D modelov iz barvnih slik v grobem delimo na redko in gosto rekonstrukcijo. Najbolj popularen postopek za prvo je imenovan struktura iz gibanja (angl. structure from motion) označen s kratico SfM [3]. Sestavljen je lahko iz različnih algoritmov, v splošnem pa deluje tako, da se na slikah poiščejo značilnice, nato pa se na podlagi ujemanj med slikami izračunajo njihove 3D koordinate. Rezultat je redek oblak 3D točk ter pozicija in orientacija kamer, ki pripadajo vhodnim slikam (skupaj imenovano tudi redka rekonstrukcija). Ureditev vhodnih slik je lahko naključna.

V zelo znanem delu [4] so avtorji takšen postopek uporabili za interaktivno brskanje po velikih neurejenih zbirkah fotografij. Njihov rezultat je t. i. 3D brskalnik, ki iz množice vhodnih slik zgradi redko rekonstrukcijo in posledično določi lokacije pripadajočih kamer. Tako se lahko uporabnik sprehaja po navideznem prostoru, išče slike specifičnega objekta itd. Pridobljene rekonstrukcije v tem delu vsebujejo nekaj 100 slik. V delu [5] so avtorji z izvirno strategijo uporabe algoritmov pokazali, da je lahko časovna zahtevnost postopka SfM skoraj linearna tudi pri rekonstrukciji z več tisoč slikami. Dejanska časovna zahtevnost sicer ni linearna, vendar so koeficienti višjih členov zelo majhni. Na takšnem postopku temeljijo tudi številni sodobni pristopi za redko rekonstrukcijo, uporabljamo pa ga tudi v našem delu. Omembe vredno je še delo [6], kjer so avtorji v nekaj dneh pridelali rekonstrukcije večjih mest iz neurejene množice slik pridobljenih s spleta. Rekonstrukcije so izgrajene iz več 10000 slik.

Če želimo na podlagi slik pridobiti 3D model visoke ločljivosti, ki se bolj natančno prilega obliki dejanskega predmeta, potem lahko redek oblak točk nadgradimo z gosto rekonstrukcijo. Za razliko od redke rekonstrukcije je v tem primeru cilj pridobiti globino (in posledično 3D lokacijo) vsakega slikovnega elementa na vsaki sliki. To storimo z algoritmi s področja večpoglednega sterea (angl. multi-view stereo), ki za vhod vzamejo rezultate redke rekonstrukcije in proizvedejo gost 3D model. Opis in primerjava različnih pristopov rekonstrukcije na tem področju je na voljo v preglednem članku [7].

Na tem mestu naj omenimo še sorodni postopek imenovan vizualni SLAM (angl. visual simultaneous localization and mapping). Kljub temu, da je njegov prvotni namen lokalizacija robota, lahko nekatere sodobnejše implementacije proizvedejo redke [8] in goste [9] rekonstrukcije. Takšne rešitve so zaenkrat pri gradnji 3D modelov večinoma manj robustne in fleksibilne v primerjavi s strukturo iz gibanja.

V prejšnjem podpoglavju smo že omenili, da mora biti uporabnik pri zajemanju slik pozoren na nekatere predpostavke in omejitve algoritmov za rekonstrukcijo. Ta problem je naslovljen tudi v delu [2]. Avtorji so razvili rešitev, ki uporabniku pomaga s sprotno informacijo o ustreznosti slik in kvaliteti modela. V delu [10] je bil ta sistem uporabljen pri rekonstrukciji z uporabo kvadrokopterjev. Poleg tega so avtorji na prizorišče rekonstrukcije postavili posebno pripravljene vizualne oznake, ki omogočajo avtomatsko kalibracijo kamere, določanje absolutne velikosti rekonstrukcije in umestitev modela v referenčni geografski koordinatni sistem. Kvadrokopter je bil voden s strani uporabnika. V delu [11] so avtorji uporabili obogateno resničnost za podporo pri navigaciji kvadrokopterja. Uporabnik lahko preko vmesnika postavi navigacijske točke in nadzira pot letenja. Povsem avtonomna rekonstrukcija s kvadrokoperjem je še vedno stvar aktivnega razvoja.

Načrtovanje najboljšega naslednjega pogleda je še en pomemben gradnik pri avtonomni rekonstrukciji, uporablja pa se lahko tudi kot pomoč uporabniku pri zajemanju slik. V literaturi so z istim pojmom označeni zelo različni pristopi, največkrat pa je načrtovanje pogledov uporabljeno v kombinaciji z robotom. V delu [12] je bil uporabljen industrijski robotski manipulator z laserskim skenerjem. Najboljši pogledi so izbrani na podlagi robov v modelu in cenovne funkcije, ki strmi k raziskovanju in čim boljši kvaliteti modela. Še en starejši postopek za načrtovanje pogledov pri laserskem skeniranju je opisan v [13]. Zanimiv pristop pa so izbrali tudi v [14], kjer je bila pri rekonstrukciji uporabljena globinska kamera in robotski manipulator ki lahko predmet obrača in preprijema. Obe omenjeni robotski roki lahko povsem avtonomno rekonstruirata predmete. Pomanjkljivost obeh sistemov je, da

se robotska roka ne more prosto premikati po prostoru in sta zato omejena na rekonstrukcijo manjših predmetov. Poleg tega so uporabljeni roboti in senzorji, ki so za širšo uporabo lahko predragi.

Načrtovanje najboljšega naslednjega pogleda na področju strukture iz gibanja so naslovili v članku [15]. Predlagana je cenovna funkcija, ki združuje negotovost v oceni strukture, projekcijo modela in videz teksture. Funkcija je optimizirana z algoritmom Nelder-Mead [16], ki za optimizacijo ne potrebuje odvodov.

Eni izmed bolj priljubljenih obstoječih odprtokodnih rešitev za gradnjo 3D modelov iz množice slik sta VisualSFM [17] in COLMAP [18]. Ti rešitvi združujeta algoritme redke in goste rekonstrukcije ter ponujata uporabniški vmesnik. Obstajajo tudi sorodne komercialne rešitve, ki ponujajo bolj dodelano uporabniško izkušnjo in večji nabor funkcionalnosti. Med bolj priljubljenimi so na primer: 3DF Zephyr [1], Agisoft Metashape [2] in Reality Capture [3]. Vse omenjena rešitve delujejo tako, da uporabnik predhodno zajame slike in jih nato uvozi v program, ki proizvede 3D model. Naša rešitev za razliko od omenjenih ponuja interaktivno izkušnjo, ki s prikazom trenutnega stanja rekonstrukcije, ocene kvalitete modela in predlaganjem naslednjega pogleda nudi uporabniku dodatne informacije in pomoč pri zajemanju slik.

[1] https://www.3dflow.net/3df-zephyr-pro-3d-models-from-photos/
[2] https://www.agisoft.com/
[3] https://www.capturingreality.com/

1.3 Prispevki

V tem delu načrtujemo naslednje prispevke:

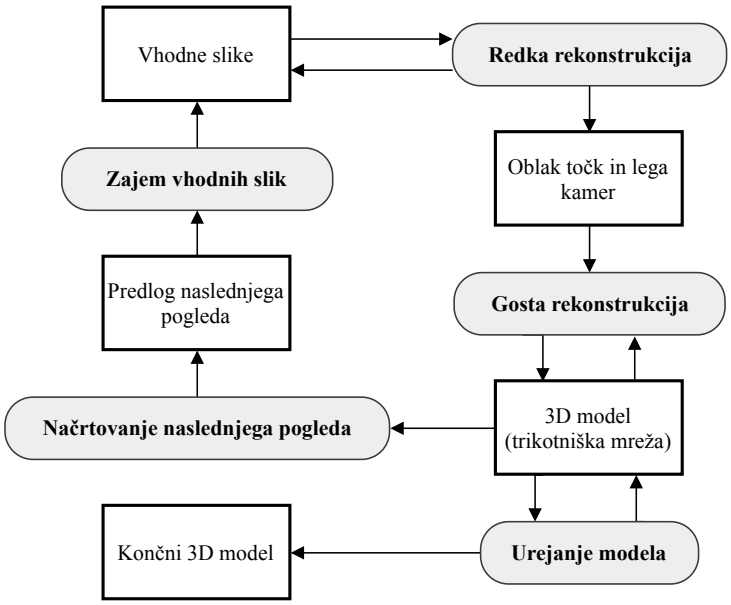
- Razvoj postopka in programske opreme, ki podpira celoten proces rekonstrukcije iz barvnih slik z naslednjimi funkcionalnostmi: zajemanje in dodajanje slik k rekonstrukciji preko spletne ali IP kamere, lokalizacija oz. določanje trenutne lokacije kamere glede na model (na podlagi zajete slike), prikaz trenutnega 3D modela in njegove kvalitete, predlaganje najboljšega naslednjega pogleda, osnovno urejanje modela in tvorjenje končnega 3D modela s teksturo. Pri implementaciji bomo uporabili obstoječe odprtokodne knjižnice. Tekoče delovanje je predvideno za rekonstrukcije iz približno 100 slik.

- Zasnova in implementacija mere za oceno kvalitete 3D modela brez poznavanja referenčnega modela.

- Razvoj in implementacija metode za načrtovanje najboljšega naslednjega pogleda, ki temelji na omenjeni meri kvalitete 3D modela.

1.4 Oris rešitve

Grob oris programske rešitve razvite v tem delu je prikazan na sliki 1.1. Diagram prikazuje odvisnost med funkcionalnostmi programa ter vhodnimi in izhodnimi podatki. Postopek se prične z zajemanjem slik, ki so pridobljene preko IP kamere. Iz njih je postopoma grajena redka rekonstrukcija, ki proizvede oblak točk. Po vsaki dodani sliki je na podlagi trenutnega oblaka točk rekonstruirana tudi površina. Tako pridobljeni 3D model je predstavljen s trikotniško mrežo. Na podlagi te predstavitve so predlagani tudi naslednji pogledi. Ko je zajetih dovolj slik, lahko uporabnik z dodatnim urejanjem, izboljšavo ločljivosti in teksturiranjem pridobi končni 3D model. Celoten sistem za rekonstrukcijo je podrobneje predstavljen v poglavju 5.



Slika 1.1: Grob oris programske rešitve. Sivi okvirji predstavljajo funkcionalnosti programa, beli pa njihove vhode in izhode. Puščice prikazujejo potek rekonstrukcije in odvisnost med funkcionalnostmi.

1.5 Struktura dela

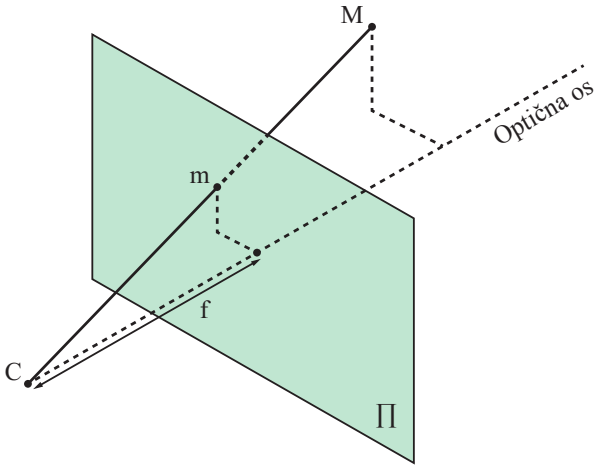
V poglavju 2 opišemo algoritme uporabljene pri redki rekonstrukciji, katere rezultat je oblak točk ter pozicija in orientacija kamer. V poglavju 3 opišemo algoritem za tvorjenje površine in ostale algoritme, ki proizvedejo gost in teksturiran 3D model. Ti algoritmi kot vhod sprejmejo rezultate redke rekonstrukcije. V poglavju 4 opišemo uporabljeno mero za oceno kvalitete modela in predstavimo postopek za načrtovanje najboljšega naslednjega pogleda. V poglavju 5 opišemo podrobnosti implementacije, predstavimo uporabljene knjižnice in utemeljimo njihovo izbiro. V poglavju 6 opišemo način eksperimentalnega vrednotenja ter rezultate našega sistema za rekonstrukcijo in napovedovanje naslednjih pogledov. V poglavju 7 podamo sklepne ugotovitve in navedemo možnosti za nadaljnje izboljšave.

Poglavje 2

Redka rekonstrukcija

2.1 Model kamere in epipolarna geometrija

Za začetek si poglejmo nekaj teoretičnih osnov na katerih temelji metoda rekonstrukcije uporabljena v našem delu. V računalniškem vidu je kamera pogosto modelirana kot kamera s točkasto odprtino (angl. pinhole camera). Gre za preprost model, ki točke preslika iz 3D prostora na 2D ravnino preko samo enega žarka, kot je to prikazano na sliki 2.1.



Slika 2.1: Projekcija 3D točke M na slikovno ravnino Π. Rezultat projekcije je 2D točka m. Center kamere je v točki C, f pa predstavlja goriščno razdaljo.

Parametri modela so razdeljeni na notranje, ki opišejo projekcijo za kamero v izhodišču koordinatnega sistema, in zunanje, ki opišejo njeno pozicijo in orientacijo. Najpomembnejši med notranjimi je goriščna razdalja f, ki predstavlja razdaljo med centrom kamere in projekcijsko ravnino. Preostali parametri so posledica majhnih nepravilnosti pri izdelavi kamer. Ti so odmik optične osi od središča slikovne ravnine c_x in c_y, razmerje dolžine stranic slikovnih elementov a ter koeficient poševnosti koordinatnih osi projekcijske ravnine s. Pri sodobnih kamerah so privzete vrednosti blizu dejanskim, njihovo bolj natančno oceno pa lahko pridobimo s postopkom kalibracije kamere [19]. Notranje parametre zapišemo s 3×3 kalibracijsko matriko

$$K = \begin{bmatrix} f & s & c_x \\ 0 & f\ a & c_y \\ 0 & 0 & 1 \end{bmatrix}. \tag{2.1}$$

Pomembni so tudi nelinearni notranji parametri ki modelirajo distorzijo leče. Ker jih ne moremo vključiti v preprost linearni model, jih ponavadi obravnavamo ločeno. Radialna distorzija je posledica oblike leče, ki je zaradi lažje izdelave sferična, namesto bolj idealne parabolične. Distorzija je aproksimirana s polinomom šeste stopnje, ki zadostuje za veliko večino leč. Enačbi (2.2) opisujeta distorzijo originalne točke (x, y) v deformirano (x_{rad}, y_{rad}), kjer je $r = \sqrt{(x^2 + y^2)}$ razdalja originalne točke do izhodišča. Modeliramo tudi tangencialno distorzijo in sicer z enačbama (2.3). Pojavi se, ker leča in slikovna ravnina nista povsem vzporedno poravnani.

$$\begin{aligned} x_{rad} &= x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \\ y_{rad} &= y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \end{aligned} \tag{2.2}$$

$$\begin{aligned} x_{tan} &= x + (2p_1 xy + p_2(r^2 + 2x^2)) \\ y_{tan} &= y + (p_1(r^2 + 2y^2) + 2p_2 xy) \end{aligned} \tag{2.3}$$

Distorzijo leče torej modeliramo z dodatnimi petimi notranjimi parametri k_1, k_2, k_3, p_1 in p_2. Če je distorzija na slikah majhna, lahko nekatere tudi izpustimo. Opisani model je pogosto uporabljen in je podrobneje opisan v [20].

Zunanji parametri opišejo pozicijo in orientacijo kamere v 3D prostoru oz. preslikavo koordinatnega sistema iz 3D prostora v koordinatni sistem kamere. Skupnih šest parametrov je predstavljenih s 3×1 translacijskim vektorjem t in 3×3 rotacijsko matriko R, za katero veljajo določene omejitve in ima v resnici samo tri proste parametre. Parametri so združeni v afino transformacijsko matriko C

$$C = \Big[R \mid t \Big]; \quad R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}, \quad t = \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}. \tag{2.4}$$

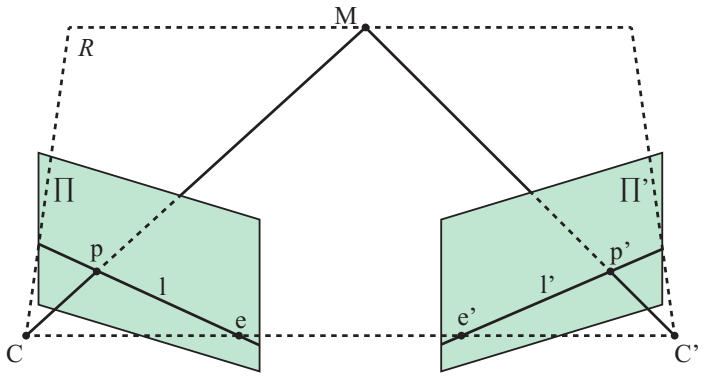
Projekcija 3D točke M v 2D točko m je na ta način lahko predstavljena kot preprosto množenje matrik, kot to prikazuje enačba (2.5). Točke so tu zapisane v homogenih koordinatah.

$$m = K\ C\ M; \quad m = \frac{1}{z} \begin{bmatrix} x \\ y \\ z \end{bmatrix}, \quad M = \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{2.5}$$

V kontekstu rekonstrukcije želimo za vsako sliko poiskati pripadajoče zunanje parametre, za notranje parametre pa lahko predpostavimo da so fiksni, če so slike zajete z isto kamero.

Na rekonstrukcijo lahko gledamo kot obraten postopek zajemanja slike. Iz 2D projekcije želimo namreč nazaj pridobiti 3D lego točk v prostoru. Da lahko to dosežemo potrebujemo vsaj dva pogleda na katerih je vidnih več istih 3D točk. Razmerje med takšnim parom pogledov opisuje epipolarna geometrija, ki jo prikazuje slika 2.2. Prikazan je sistem dveh kamer C in C'. 3D Točka M se nahaja v prostoru in je vidna v obeh kamerah. Preslika se na projekcijski ravnini Π in Π' v točki p in p'. Vse omenjene točke ležijo na isti ravnini R imenovani epipolarna ravnina. Epipolarni premici l in l' sta presečišči med R ter ravninama Π in Π'. Presečišči med premico, ki povezuje oba centra projekcije C in C', ter ravninama Π in Π' sta epipola e in e'.

Epipolarno geometrijo opisuje 3×3 matrika F imenovana fundamentalna matrika (angl. fundamental matrix). Uporabna je pri iskanju korespondenc



Slika 2.2: Sistem dveh kamer C in C' ter epipolarna geometrija, ki opisuje njuno razmerje. 3D točka M se preslika na projekcijski ravnini Π in Π' v točki p in p', ki ležita na pripadajočih epipolarnih premicah l in l'.

med točkami na sliki. Korespondenca točke p mora namreč ležati na pripadajoči epipolarni premici l', kar lahko z enačbo zapišemo kot $l' = Fp$. Na tej podlagi uvedemo pomembno epipolarno omejitev, ki jo zapišemo z enačbo

$$p^{\mathsf{T}} F p' = 0. \tag{2.6}$$

Fundamentalna matrika zajema informacije o rotaciji, translaciji in notranjih parametrih, ki povezujejo par kamer. Če so notranji parametri poznani, lahko namesto nje uporabimo osnovno matriko E (angl. essential matrix), ki jo pridobimo z enačbo

$$E = K^{\mathsf{T}} F K. \tag{2.7}$$

Iz nje lahko izpeljemo orientacijo druge kamere glede na prvo in smer translacije. Absolutne dolžine translacije ne moremo določiti samo na podlagi korespondenc, zato pravimo da je rekonstrukcija določena do velikosti natančno. Točka M se nahaja na presečišču premic Cp in $C'p'$. Tako lahko na podlagi epipolarne geometrije in korespondenc med točkami nazaj pridobimo njihovo lego v 3D prostoru.

2.2 Oris postopka

Gradnja končnega 3D modela je razdeljena na redko in gosto rekonstrukcijo. Prva je opisana v tem poglavju, druga pa v poglavju 3. Redka rekonstrukcija iz vhodnih slik (oz. pogledov) proizvede oblak 3D točk ter oceni pozicije in orientacije kamer v 3D prostoru. Gosta rekonstrukcija sprejme te rezultate kot vhod in proizvede 3D model visoke ločljivosti s površino in teksturo.

V našem primeru smo za redko rekonstrukcijo iz barvnih slik (v tem poglavju rekonstrukcija) uporabili postopek imenovan *inkrementalna struktura iz gibanja* [3] (v nadaljevanju struktura iz gibanja). Deluje tako, da se vzpostavi začetna rekonstrukcija na podlagi prvih dveh pogledov, ostali pogledi pa so inkrementalno dodani k rekonstrukciji. Obstaja še alternativni postopek imenovan globalna struktura iz gibanja, ki deluje tako, da se lege kamer določajo hkrati [21]. Ta postopek je namenjen predvsem odloženemu procesiranju in v našem primeru ne pride v poštev, saj želimo, da uporabnik dobi sprotni odziv pri dodajanju vsake slike.

Struktura iz gibanja je sestavljena iz več korakov. Nalogo vsakega izmed njih lahko opravimo z različnimi algoritmi, ki imajo svoje prednosti in slabosti. Tu bomo opisali splošen postopek, ki je uporabljen v tem delu in sledi tistemu opisanem v članku [5], konkretni algoritmi pa so opisani v nadaljnjih podpoglavjih.

Osnova za rekonstrukcijo je iskanje ujemanj med slikovnimi elementi. Paru slikovnih elementov na različnih slikah pravimo da se ujemata, če predstavljata isto točko v prostoru. Direktno iskanje ujemanj med vsemi slikovnimi elementi je preveč časovno zahtevno in premalo robustno, zato je pomemben gradnik pri številnih metodah računalniškega vida pridobivanje značilnic. Značilnice so izrazite točke na sliki z razpoznavno okolico, ki nam omogoča njihovo ponovljivo iskanje. Predstavitev slik je tako iz slikovnih elementov zreducirana na manjšo in bolj obvladljivo množico točk, ki so bolj robustne pri iskanju ujemanj. V našem primeru uporabljamo značilnice SIFT [22], ki so invariantne na skaliranje, rotacijo, osvetljenost in manjše perspektivne spremembe.

Rekonstrukcijo R inicializiramo na podlagi prvih dveh slik I_1 in I_2. Če kamera ni predhodno kalibrirana, potem za notranje parametre uporabimo začetno oceno in jih med rekonstrukcijo optimiziramo. Ujemanja značilnic med I_1 in I_2 nam omogočajo, da ocenimo pripadajočo matriko E in posledično pridobimo relativno pozicijo in orientacijo prvih dveh kamer. S triangulacijo ujemajočih značilnic vzpostavimo začetni oblak 3D točk. Rekonstrukcijo R nato postopoma nadgrajujemo z dodajanjem novih slik. Pri dodajanju nove slike I_i moramo ugotoviti s katerimi značilnicami na obstoječih slikah se ujemajo značilnice nove slike. Naivno iskanje ujemanj med vsemi pari $\{\{I_i, I_j\} \mid 1 \leq j < i\}$ je prepočasno in neskalabilno, zato ga pohitrimo z uporabo dveh hitrejših metod. Najprej med obstoječimi slikami poiščemo tiste, ki so vizualno najbolj podobne novi. To storimo s pomočjo vizualnega slovarja (angl. visual vocabulary), ki sicer konceptualno izvira s področja analize besedil, na področju računalniškega vida, pa je prilagojen iskanju podobnih slik [23]. Vzamemo najboljših c kandidatov in nato iščemo ujemanja značilnic samo med pari $\{\{I_i, I_k\} \mid k \in kandidati\}$. Iskanje ujemanj za par $\{I_i, I_k\}$ je izvedeno s postopkom, ki ga avtorji poimenujejo kaskadno zgoščevanje [24] (angl. cascade hashing). Temelji na lokalno občutljivih zgoščevalnih funkcijah, ki podobne značilnice preslikajo v iste binarne vrednosti in jih tako implicitno gručijo. Ujemanja lahko nato iščemo znotraj gruče, kar močno zmanjša število potrebnih primerjav. S to metodo lahko dosežemo pohitritev za dva reda velikosti v primerjavi z naivnim pristopom. Ker značilnice zajemajo samo lokalno informacijo o sliki, ta korak proizvede tudi napačna ujemanja. Filtriramo jih z uveljavljanjem epipolarne omejitve, tako da ohranimo čim večjo množico ujemanj za katere velja enačba (2.6). To storimo s pomočjo algoritma RANSAC [25], ki v splošnem omogoča robustno ocenjevanje parametrov modela na šumnih podatkih.

Značilnice nove slike, za katere smo našli ujemanje na vsaj eni izmed obstoječih slik, lahko sedaj razdelimo v dve množici. V množici A so tiste, ki so že del rekonstrukcije R, v množici B pa preostale, ki imajo ujemanja, ampak še niso rekonstruirane. Iz množice A vzpostavimo 2D-3D korespondence

med I_i in R. Že tri takšne korespondence so dovolj da določimo absolutno lego kamere v prostoru rekonstrukcije R, za bolj robustno oceno pa si jih želimo čim več. Med podatki se pojavijo tudi napačne korespondence, zato algoritem ponovno uporabimo v kombinaciji z algoritmom RANSAC. Ko je lega nove kamere poznana, rekonstrukciji R s triangulacijo dodamo še točke iz množice B. Postopek ponovimo za vsako sliko, ki jo želimo dodati.

Algoritem 1 Struktura iz gibanja

Vhod: Sekvenca barvnih slik $\boldsymbol{I} = \{I_1, I_2, \ldots, I_n\}$

Izhod: Oblak 3D točk M ter lega kamer C oz. rekonstrukcija $\boldsymbol{R}$

1: $Z_1, Z_2 \leftarrow$ Pridobivanje značilnic SIFT iz I_1 in I_2
2: $U_{12} \leftarrow$ Ujemanja med Z_1 in Z_2
3: $E_{12} \leftarrow$ Osnovna matrika izračunana iz U_{12}
4: $C \leftarrow$ Relativna lega kamer C_1 in C_2 iz osnovne matrike E_{12}
5: $M \leftarrow$ Inicializacija oblaka 3D točk s triangulacijo U_{12}
6: **for** $i \leftarrow 3 \ldots n$ **do**
7: $\quad Z_i \leftarrow$ Pridobivanje značilnic SIFT iz I_i
8: $\quad \{I_k \mid k \in kand\} \leftarrow$ Najboljših $c = 5$ kandidatov za ujemanje
9: $\quad \{U_{ik} \mid k \in kand\} \leftarrow$ Ujemanja značilnic med Z_i in kandidati
10: $\quad A, B \leftarrow$ Implicitna delitev značilnic $Z_i \cap \{U_{ik} \mid k \in kand\}$
11: $\quad P_i \leftarrow$ Vzpostavitev 2D-3D korespondenc med Z_i in $\boldsymbol{R}$ na podlagi A
12: $\quad C_i \leftarrow$ Absolutna lega kamere za sliko I_i na podlagi P_i
13: $\quad M_i \leftarrow$ Triangulacija točk iz množice B na podlagi C_i
14: $\quad$ **if** M večji za $p = 5$ odstotkov **then**
15: $\quad\quad M, C \leftarrow$ Globalna minimizacija reprojekcijske napake
16: $\quad$ **else**
17: $\quad\quad M, C \leftarrow$ Delna minimizacija napake za zadnjih $m = 20$ slik
18: $\quad$ **end if**
19: **end for**
20: $M = \bigcup_{i=1}^{n} M_i$ in $C = \bigcup_{i=1}^{n} C_i$
21: $\boldsymbol{R} = \{M \cup C\}$

Natančnost rekonstrukcije izboljšamo z minimizacijo reprojekcijske napake, ki je definirana kot seštevek razdalj med projekcijo 3D točke na vse slike na katerih je vidna in lokacijo pripadajočih značilnic na slikah. Ker gre za časovno zahteven postopek, globalno minimizacijo izvedemo samo, če se število točk v rekonstrukciji poveča za več kot p odstotkov, sicer po dodajanju vsake slike izvedemo delno minimizacijo, ki optimizira 3D točke in poglede samo zadnjih m slik. Celoten postopek je povzet v algoritmu 1. Za spremenljivke omenjene v besedilu so podane tudi uporabljene vrednosti. V nadaljevanju tega poglavja so bolj podrobno predstavljeni koraki postopka.

2.3 Pridobivanje značilnic

Omenili smo že, da s pridobivanjem značilnic slikovne elemente slike transformiramo v manjšo in bolj obvladljivo množico robustnih točk. Obstaja veliko število različnih algoritmov za pridobivanje značilnic, ki imajo svoje prednosti in slabosti. V našem delu smo uporabili verjetno najbolj znane značilnice SIFT [22]. Učinkovitost nekaterih alternativnih algoritmov je na področju rekonstrukcije primerjana v [26].

Postopek pridobivanja je razdeljen v štiri glavne korake. Preden začnemo, originalno sliko pretvorimo v sivinsko, podvojimo njeno velikost z bilinearno interpolacijo in jo gladimo z Gaussovim filtrom ($\sigma_n = 0{,}5$). Rezultat je slika I_{σ_n}. Glajenje opisuje enačba (2.8), kjer $L(x, y, \sigma)$ predstavlja rezultat glajenja slike s konvolucijskim jedrom $G(x, y, \sigma)$. Razni parametri v nadaljevanju so nastavljeni na takšne vrednosti, kot so predlagane v originalnem članku.

$$\begin{aligned} G(x, y, \sigma) &= \frac{1}{2\pi\sigma^2}\, e^{-\left(x^2+y^2\right)/2\sigma^2} \\ L(x, y, \sigma) &= G(x, y, \sigma) * I(x, y) \end{aligned} \tag{2.8}$$

Prvi korak je detekcija ekstremov v prostoru velikosti (angl. scale space). Velikost v našem kontekstu predstavlja parameter glajenja σ. Konstrukcija prostora velikosti je naslednja. Iz začetne slike I_{σ_n} pridelamo $S = 6$ novih glajenih slik. Prva je glajena z osnovnim parametrom $\sigma_0 = 1{,}6$, vsaka

naslednja pa s parametrom, ki je za faktor $k = \sqrt{2}$ večji od predhodnega. Skupini pridobljenih slik pravimo oktava. Naslednjo oktavo pridobimo tako, da prepolovimo velikost zadnje slike prejšnje oktave in ponovimo glajenje. Postopek ponavljamo dokler ne pridobimo želenega števila oktav $O = 4$. Rezultat je prostor velikosti s katerim dosežemo invariantnost značilnic na velikostne spremembe. Za iskanje ekstremov bi v idealnem primeru uporabili Laplaceov operator na glajeni sliki (angl. Laplacian of Gaussian), vendar je računanje drugih odvodov računsko preveč intenzivno. Ta operator zato aproksimiramo z razliko glajenih slik različnih velikosti (angl. Difference of Gaussians), kot to opisuje spodnja enačba

$$D(x, y, \sigma) = L(x, y, k_i\sigma) - L(x, y, k_j\sigma). \tag{2.9}$$

Operator $D(x, y, \sigma)$ uporabimo na sosednjih slikah znotraj oktav in tako pridobimo nove slike DoG. Ekstreme poiščemo na slikah DoG, tako da vsak slikovni element primerjamo z vsemi 26 sosedi (8 na isti sliki in preostalih 18 na sosednjih slikah znotraj oktave). Robnih slikovnih elementov ne upoštevamo, ker nimajo vseh sosedov. Če je vrednost večja ali manjša od okolice, potem gre za ekstrem. Na sliki 2.3 je prikazan prostor velikosti za slike DoG in ekstremi, ki so rezultat opisanega postopka.

Drugi korak je natančna lokalizacija značilnic. Ekstremi pridobljeni v prejšnjem koraku so uporabljeni kot začetni približek, želimo pa si natančnosti, ki je večja od diskretnih lokacij slikovnih elementov. Okolico ekstrema uporabimo za aproksimacijo Taylorjevega polinoma druge stopnje, ki ga zapišemo z enačbo (2.10). Vektor $\mathbf{x} = (x, y, \sigma)^\mathsf{T}$ predstavlja odmik od ekstrema. Natančnejšo lokacijo pridobimo tako, da polinom odvajamo, enačimo z nič in rešimo enačbo

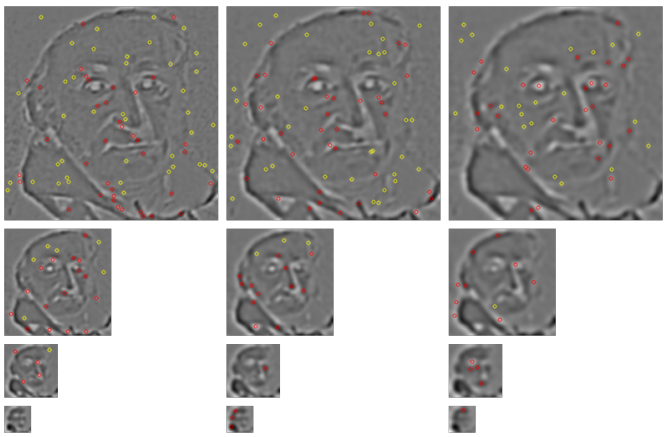
$$D(\mathbf{x}) = D + \frac{\partial D^\mathsf{T}}{\partial \mathbf{x}}\mathbf{x} + \frac{1}{2}\mathbf{x}^\mathsf{T}\frac{\partial^2 D}{\partial \mathbf{x}^2}\mathbf{x}. \tag{2.10}$$

V tem koraku se še znebimo manj zanesljivih ekstremov. Najprej preverimo absolutno vrednost ekstrema. Če je ta manjša od $t_p = 0{,}03$ ga odstranimo, saj je verjetno posledica šuma. Znebimo se še tistih, ki so posledica robov. Za vsak ekstrem pridobimo Hessovo matriko H, ki zajema ukrivljenost funkcije

v njegovi okolici. Nato z enačbo izračunamo razmerje r

$$r = \frac{\mathrm{tr}(H)^2}{\det(H)}, \qquad H = \begin{bmatrix} \frac{\partial^2 \mathrm{D(x,y,\sigma)}}{\partial x^2} & \frac{\partial^2 \mathrm{D(x,y,\sigma)}}{\partial x \partial y} \\ \frac{\partial^2 \mathrm{D(x,y,\sigma)}}{\partial x \partial y} & \frac{\partial^2 \mathrm{D(x,y,\sigma)}}{\partial y^2} \end{bmatrix}. \tag{2.11}$$

Obdržimo samo ekstreme za katere je r manjši od $(t_e + 1)^2 / t_e$, kjer je $t_e = 10$. Na ta način izboljšamo stabilnost in iskanje ujemanj med značilnicami.

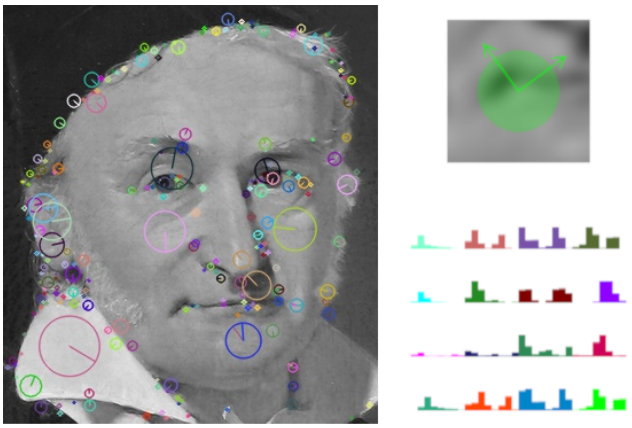


Slika 2.3: Slika prikazuje konstrukcijo prostora velikosti za parametra $O = 4$ in $S = 6$. Vrstice predstavljajo oktave, po stolpcih v desno pa narašča velikost σ. Začnemo s 6 glajenimi slikami, iz katerih pridobimo 5 slik DoG, ekstreme pa poiščemo na treh sredinskih slikah ki imajo vse sosede. Rumeni ekstremi so odstranjeni v sledečih korakih, rdeči pa so dovolj zanesljivi [27].

V tretjem koraku ekstremom dodelimo orientacijo. Najprej za slikovne elemente na sliki $L(x, y, \sigma)$ v okolici ekstrema izračunamo magnitudo $m(x, y)$ in smer gradienta $\theta(x, y)$ z enačbama (2.12), kjer je $L(x, y)$ slikovni element na sliki. Za okolico štejemo vse slikovne elemente znotraj Gaussovega okna velikosti 1,5σ.

$$\begin{aligned} m(x,y) &= \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2} \\ \theta(x,y) &= \tan^{-1}\left(\frac{L(x,y+1) - L(x,y-1)}{L(x+1,y) - L(x-1,y)}\right) \end{aligned} \tag{2.12}$$

Nato ustvarimo histogram smeri s 36 razdelki (vsak pokriva razpon 10°). Vsakemu vzorcu iz okolice je dodeljen razdelek histograma glede na njegovo smer gradienta $\theta(x, y)$. Pripadajočemu razdelku vzorec prišteje svojo magnitudo gradienta $m(x, y)$. Ta je dodatno utežena z Gaussovim oknom s središčem v ekstremu (tako zmanjšamo prispevek bolj oddaljenih vzorcev). Značilnici se določi smer z največjo vrednostjo v histogramu. Če obstaja še kakšna vrednost, ki je večja od 80% maksimalne, se ustvari nova značilnica z enako lokacijo in velikostjo, za orientacijo pa vzamemo drugo vrednost.



Slika 2.4: Na levi strani je prikazano pridobivanje značilnic na konkretnem primeru. Prikazane so njihove orientacije in velikosti. Na desni strani je grafični prikaz opisnika za eno izmed značilnic. Zeleni osi predstavljata poravnavo koordinatnega sistema z orientacijo značilnice. Spodaj so prikazani histogrami za vsako izmed 4×4 regij [27].

Točke z lokacijo, velikostjo in orientacijo imenujemo značilnice. V zadnjem koraku jim pripišemo še opisnik (angl. descriptor). To je vektor s 128 vrednostmi, ki zajema informacije o njihovi okolici. Uporablja se pri medsebojnem primerjanju značilnic in iskanju ujemanj. Opisnik značilnice pridobimo iz okolice slikovnih elementov velikosti 16×16 oz. njihovih magnitud in smeri gradienta. Za začetek od smeri gradienta odštejemo orientacijo značilnice. Tako dosežemo rotacijsko neodvisnost opisnika. Nato okolico enakomerno razdelimo na manjše regije velikosti 4×4. Sedaj podobno kot

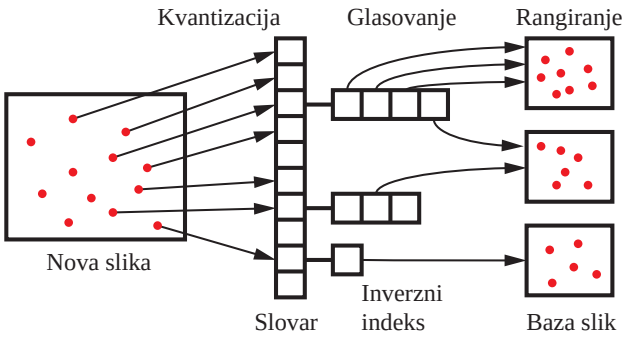
v prejšnjem koraku za vsako regijo zgradimo histogram smeri, tokrat z 8 razdelki (vsak pokriva razpon 45°). Vsak vzorec v regiji prišteje svojo magnitudo gradienta pripadajočemu razdelku. Magnitude so ponovno utežene z Gaussovim oknom velikosti $1{,}5\sigma$. Vrednosti dobljenih histogramov združimo v vektor velikosti $4 \times 4 \times 8 = 128$. Tako pridobimo opisnik značilnice. Da je opisnik bolj odporen na spremembe osvetlitve ga normaliziramo, obrežemo vrednosti ki presegajo 0,2 in ga ponovno normaliziramo. Končni rezultat pridobivanja značilnic je prikazan na levi strani slike 2.4. Na desni je grafični prikaz opisnika. Ko v preostalih poglavjih govorimo o značilnicah, s tem mislimo tudi na pripadajoči opisnik.

2.4 Ujemanja med značilnicami

Ujemanja med značilnicami so osnova za vzpostavitev in razširitev rekonstrukcije. V našem primeru pride časovna zahtevnost tega problema do izraza predvsem pri dodajanju nove slike k rekonstrukciji. Naš cilj je namreč, da za značilnice nove slike poiščemo ujemanja z značilnicami obstoječih slik, ki so že del rekonstrukcije. Z naivnim pristopom bi ujemanja iskali z vsemi predhodno dodanimi slikami, kar je prepočasno že pri rekonstrukcijah skromne velikosti. Postopek zato pohitrimo z dvema hitrejšima pristopoma. Najprej s pomočjo vizualnega slovarja med obstoječimi slikami poiščemo kandidate za ujemanje. Dejansko iskanje ujemanj značilnic med novo sliko in kandidati pa izvedemo s postopkom, ki ga avtorji imenujejo kaskadno zgoščevanje.

2.4.1 Iskanje podobnih slik

Za začetek potrebujemo vizualni slovar (v nadaljevanju slovar). Med rekonstrukcijo dodajamo slike predstavljene z vizualnimi besedami slovarja v podatkovno zbirko. Nato lahko za novo sliko naredimo poizvedbo, ki vrne podobne slike oz. kandidate za ujemanje. Takšen oris delovanja je prikazan na sliki 2.5. Za ta korak v našem delu uporabljamo knjižnico, ki je implementirana po [28]. Opis pomembnejših korakov sledi v nadaljevanju.

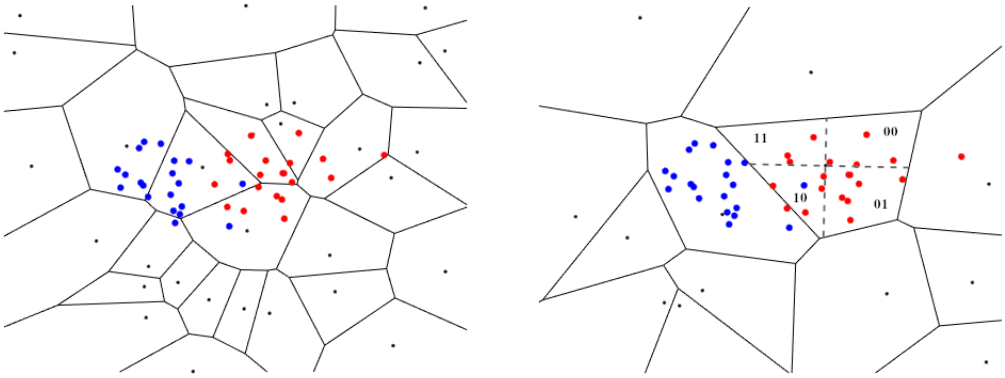


Slika 2.5: Za značilnice nove slike poiščemo najbližje besede v slovarju in preko inverznega indeksa pridobimo c najbolj podobnih slik. Med njimi in novo sliko poiščemo ujemanja, kar nam omogoča, da v naslednjem koraku pridobimo lego kamere. Skica je povzeta po [23].

Slovar je potrebno zgraditi preden ga lahko uporabimo pri rekonstrukciji. Za začetek potrebujemo veliko učno množico raznolikih slik, na katerih poiščemo značilnice. Vse pripadajoče opisnike z algoritmom k-povprečij (angl. k-means) razporedimo v k gruč. Središče gruče je nov opisnik imenovan vizualna beseda (v nadaljevanju beseda). Ta korak je imenovan kvantizacija, saj opisnike $x \in \mathbb{R}^d$ iz zveznega prostora preslikamo v indeks najbližje besede, kar je prikazano v enačbi

$$\begin{aligned} q : \mathbb{R}^d &\to [1, k] \\ x &\mapsto q(x) \ . \end{aligned} \tag{2.13}$$

Za hitro iskanje najbližjih sosedov je uporabljena drevesna struktura, zato takšnemu slovarju pravimo tudi besediščno drevo (angl. vocabulary tree). Parameter k pomembno vpliva na kvaliteto slovarja. Pri velikih vrednostih, so podobni opisniki preslikani v različne besede. Za premajhne vrednosti, pa so različni opisniki preslikani v iste besede. Ta kompromis je prikazan na sliki 2.6. V delu [29] so avtorji osnovno različico slovarja nadgradili s Hammingovo vložitvijo opisnikov. Takšen pristop združuje prednosti redkejše in gostejše kvantizacije in v splošnem daje boljše rezultate, zato ga uporabljamo tudi v našem delu.



Slika 2.6: Vizualni prikaz kvantizacije in vpliv parametra k. Leva stran $k = 30$. Desna stran $k = 13$. Točke iste barve predstavljajo šumne različice istega opisnika. Na desni strani je prikazana tudi implicitna delitev gruče pri uporabi Hammingove vložitve. Skica je povzeta po [29].

Hammingova vložitev (angl. Hamming embedding) je preslikava opisnika iz Evklidskega prostora v njegov d_b dimenzionalni binarni podpis $b_x = HE(x) = (b_1(x), \ldots, b_{d_b}(x))$. Zasnovana je tako, da Hammingova razdalja med opisnikoma x in y odraža njuno Evklidsko razdaljo. To nam omogoča, da znotraj gruče iščemo ujemanja samo med pari z nizko Hammingovo razdaljo in s tem posledično izločimo številna napačna ujemanja. Funkcija za preslikavo v Hammingov prostor HE je tudi zgrajena na podlagi učne množice, podrobnejši opis njene konstrukcije, pa je na voljo v originalnem delu [29].

Zgrajeni slovar lahko sedaj uporabimo pri rekonstrukciji. Vsako novo sliko, s katero uspešno razširimo rekonstrukcijo, dodamo v bazo obstoječih slik (ta je na začetku prazna). To storimo tako, da za opisnike poiščemo pripadajoče najbližje besede v slovarju in pripravimo njihov binarni podpis. Nad bazo slik je definiran tudi inverzni indeks, kar pomeni da vsaka beseda v slovarju kaže na slike, ki jo vsebujejo.

Naj bo $\mathcal{X}$ slika za katero želimo v bazi poiskati najbolj podobne slike. Predstavljena je z množico n opisnikov $\mathcal{X} = \{x_1, \ldots, x_n\}$. Kvantizacijska funkcija q preslika opisnik x_i v identifikator besede $q(x_i)$, tako da je $q(x_i) \in \mathcal{C}$, kjer je $\mathcal{C} = \{c_1, \ldots, c_k\}$ slovar velikosti k. Podmnožico opisnikov, ki so

preslikani v besedo c označimo z $\mathcal{X}_c = \{x \in \mathcal{X} : q(x) = c\}$. Podobnost med $\mathcal{X}$ in sliko v bazi $\mathcal{Y}$ ocenimo z enačbo

$$K(\mathcal{X}, \mathcal{Y}) = \gamma(\mathcal{X})\gamma(\mathcal{Y}) \sum_{c \in \mathcal{C}} \left(w_c M\left(\mathcal{X}_c, \mathcal{Y}_c\right)\right). \tag{2.14}$$

Normalizacijski faktor γ je definiran tako, da je samopodobnost slike enaka $K(\mathcal{X}, \mathcal{X}) = 1$. Utež w_c je določena s korenom inverzne pogostosti besede (angl. inverse document frequency). S tem preprečimo, da bi bolj pogoste besede imele prevelik prispevek pri seštevku. Enačba (2.14) računanje podobnosti porazdeli na besede, ki so skupne slikama $\mathcal{X}$ in $\mathcal{Y}$. Podobnost za posamezno besedo opisuje enačba

$$M(\mathcal{X}_c, \mathcal{Y}_c) = \sum_{x \in \mathcal{X}_c} \left(|\mathcal{Y}_c(x)|^{-1/2} \sum_{y \in \mathcal{Y}_c} f(h(b_x, b_y)) \right). \tag{2.15}$$

Funkcija h vrne Hammingovo razdaljo med binarnima podpisoma $b_x = HE(x)$ in $b_y = HE(y)$. Dobljena razdalja je vhod v utežno funkcijo f, ki je definirana z enačbo

$$f(h) = \begin{cases} e^{-h^2/\sigma^2} & \text{če } h \leq 1{,}5\sigma \\ 0 & \text{drugače} \end{cases}. \tag{2.16}$$

Z njo dosežemo, da imajo opisniki z manjšo razdaljo večji prispevek k podobnosti. V primeru da je razdalja prevelika pa prispevek izničimo. Konkretne vrednosti parametrov uporabljene v implementaciji so $d_b = 64$ (število bitov binarnega podpisa) in $\sigma = 16$. Člen $|\mathcal{Y}_c(x)|$ v enačbi (2.15) je normalizacija pojava, ki ga obravnavajo avtorji v delu [30]. Njihov argument je, da se besede, ki so prisotne na sliki, z večjo verjetnostjo pojavijo večkrat (večkratna pojavitev besed na sliki ni statistično neodvisna). Ta dodatna utež je število elementov množice $\mathcal{Y}_c$, ki so dovolj blizu x. Definirana je z $\mathcal{Y}_c(x) = \{y \in \mathcal{Y}_c : f(h(b_x, b_y)) \neq 0\}$.

Kvantizacija, drevesna struktura indeksa in uporaba inverznega indeksa nam omogočajo, da je podobnost K med $\mathcal{X}$ in vsemi slikami $\mathcal{Y}$ v bazi izračunana časovno učinkovito. Z najbolj podobnimi $c = 5$ slikami v naslednjem koraku poiščemo ujemanja med značilnicami.

2.4.2 Iskanje ujemanj

Prejšnje podpoglavje opisuje kako poiščemo slike, ki so najbolj podobne tisti, ki jo želimo dodati rekonstrukciji, to podpoglavje pa opisuje kako poiščemo ujemanja med značilnicami dveh slik. Iskanje ujemanj s primerjavo vseh možnih parov značilnic ne pride v poštev, zaradi prevelike časovne zahtevnosti. V našem delu smo zato uporabili obstoječo implementacijo algoritma predstavljenega v [24].

Najprej bolj natančno definirajmo kdaj dvema značilnicama pravimo da se ujemata. Naj bo x značilnica slike $\mathcal{X} = \{x_1, \ldots, x_n\}$, za katero iščemo ujemanje na sliki $\mathcal{Y} = \{y_1, \ldots, y_n\}$. Značilnica x se ujema s tisto značilnico y za katero je Evklidska razdalja njunih opisnikov najmanjša. V [22] je predlagan še t.i. Lowejev test, ki drastično zmanjša število napačnih ujemanj. Deluje tako, da primerjamo razdalji med najbližjo in drugo najbližjo značilnico, če je razmerje $\frac{d_1}{d_2}$ večje od mejne vrednosti 0,8 potem ujemanja ne upoštevamo.

Osnova za delovanje algoritma so lokalno občutljive zgoščevalne funkcije (angl. locality sensitive hashing). Družina takšnih funkcij $\mathcal{H}$, ki opisnik iz $\mathbb{R}^d$ slikajo v Hammingov prostor $\mathbb{B}$, je definirana z enačbama

$$\begin{aligned} &\Pr_{h\in\mathcal{H}}\left(h(x) = h(y)\right) \geq P_1, \quad \text{če } D(x,y) \leq R \\ &\Pr_{h\in\mathcal{H}}\left(h(x) = h(y)\right) \leq P_2, \quad \text{če } D(x,y) \geq cR. \end{aligned} \tag{2.17}$$

Pomen definicije $\mathcal{H}$ je naslednji. Če sta si opisnika x in y blizu, potem ju funkcija $h \in \mathcal{H}$ z večjo verjetnostjo P_1 preslika v isto vrednost. Če pa sta bolj oddaljena, potem sta v isto vrednost preslikana z manjšo verjetnostjo P_2. Namen funkcij družine $\mathcal{H}$ je torej ohraniti lokalno razmerje podatkov, obenem pa zmanjšati njihovo dimenzionalnost. V prejšnjem podpoglavju smo opisali Hammingovo vložitev s funkcijo, ki je bila uporabljena za podoben namen. Razlika je v tem, da $\mathcal{H}$ ne potrebuje faze učenja iz podatkov. Funkcija $h \in \mathcal{H}$ je določena z vektorjem r, ki je naključno vzorčen iz normalne porazdelitve $\mathcal{N}(0, I)$. Preslikavo opisnika v posamezen bit definiramo z enačbo

$$h_r(x) = \begin{cases} 1 & \text{če } r \cdot x \geq 0 \\ 0 & \text{če } r \cdot x \leq 0 \end{cases}. \tag{2.18}$$

Za daljši binarni podpis preprosto uporabimo večje število naključno vzorčenih funkcij $h \in \mathcal{H}$, rezultate pa združimo v niz bitov.

Lokalno občutljivost opisane preslikave si lahko intuitivno razlagamo na naslednji način. Vektor r v prostoru $\mathbb{R}^d$ definira hiperravnino, ki prostor naključno razdeli na dva dela. Funkcija $h_r(x)$ nam pove v kateri polovici prostora se nahaja opisnik. Daljši niz bitov pomeni večkratno delitev prostora in s tem bolj natančno določeno lokacijo opisnika. Hammingova razdalja med binarnima podpisoma različnih opisnikov je tako dobra ocena dejanske razdalje.

Algoritem kaskadnega zgoščevanja za iskanje ujemanj deluje v treh korakih, celoten postopek pa je grafično prikazan na sliki 2.7. Opisali bomo kako za en opisnik značilnice $x \in \mathcal{X}$ poiščemo najboljše ujemanje med opisniki $\mathcal{Y} = \{y_1, \ldots, y_n\}$. Prvi korak je grobo iskanje kandidatov. Za vsak opisnik proizvedemo L binarnih podpisov dolžine m

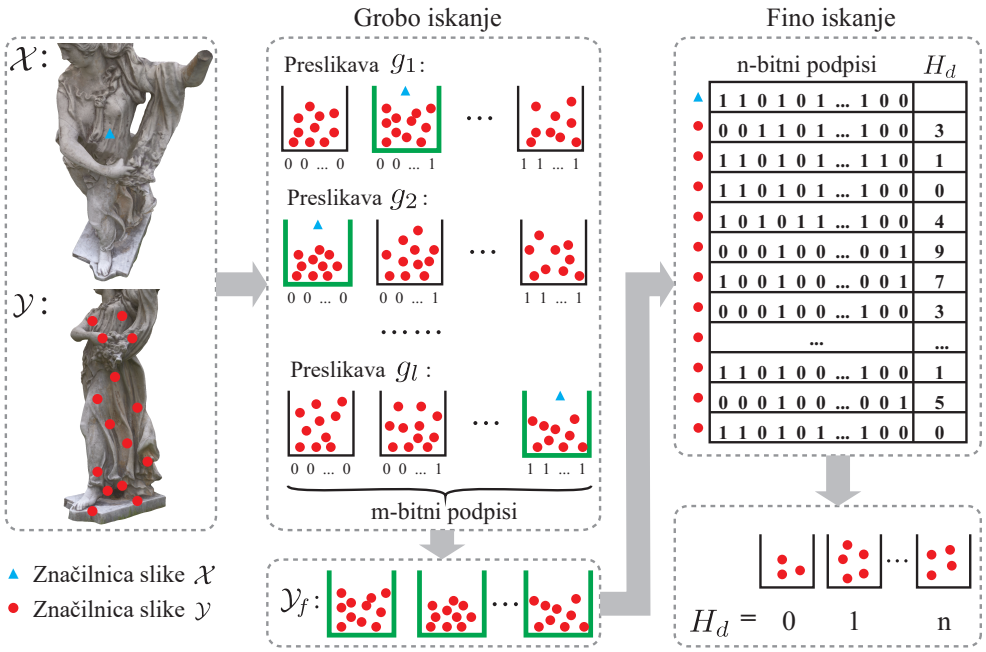
$$g_l(x) = \big(h_{1,l}(x),\ h_{2,l}(x),\ \ldots,\ h_{m,l}(x)\big); \qquad l = 1, 2, \ldots, L. \tag{2.19}$$

Funkcije $h_{i,l}$ so vzorčene neodvisno in naključno iz $\mathcal{H}$. Za nadaljnje procesiranje obdržimo samo tiste opisnike, ki se za vsaj eno izmed funkcij g_l preslikajo v isto binarno vrednost kot opisnik x. To filtrirano množico kandidatov lahko zapišemo z $\mathcal{Y}_f = \{y_i \in \mathcal{Y} : g_l(x) = g_l(y_i)\}$. Slika 2.7 prikazuje kako si lahko ta korak logično predstavljamo z zgoščenimi tabelami. Vsaka izmed funkcij g_l namreč preslika opisnike v enega izmed 2^m razdelkov. Obdržimo tiste ki so v istih razdelkih kot x. Izbira parametrov m in L je kompromis med natančnostjo in hitrostjo delovanja. Avtorji algoritma predlagajo vrednosti $m = 10$ in $L = 6$.

V drugem koraku opisnik x in kandidate $\mathcal{Y}_f$ ponovno preslikamo v Hammingov prostor. Tokrat pri preslikavi ustvarimo daljši podpis dolžine $n = 128$ bitov. Ta je pri primerjavi bolj diskriminativen od krajšega. Nov podpis je uporabljen za računanje Hammingove razdalje med x in $\mathcal{Y}_f$.

V zadnjem koraku opisnike množice $\mathcal{Y}_f$ razdelimo v skupine glede na izračunano Hammingovo razdaljo. Znotraj skupine, ki pripada najkrajši

Hammingovi razdalji, poiščemo dva opisnika z najkrajšo Evklidsko razdaljo (če je skupina z razdajo 0 prazna, preverimo naslednjo itd.). Če opisnika prestaneta Lowejev test, smo našli ujemanje.



Slika 2.7: Iskanje ujemanja za značilnico $x \in \mathcal{X}$ med značilnicami slike $\mathcal{Y}$ s kaskadnim zgoščevanjem. Grobo iskanje kandidatov opravimo z večkratno preslikavo v kratke m-bitne podpise. Za fino iskanje opisnike ponovno preslikamo, tokrat v daljše n-bitne podpise. Ujemanje poiščemo z Evklidsko razdaljo, ki jo izračunamo samo za opisnike z najkrajšo Hammingovo razdaljo H_d. Skica je povzeta po [24].

Opisniki zajemajo samo lokalno informacijo o sliki, zato ta postopek proizvede tudi napačna ujemanja. Pri rekonstrukciji je standardna praksa dodatno filtriranje z uveljavljanjem epipolarne omejitve in uporabo algoritma RANSAC. Z njim na podlagi ujemanj izračunamo epipolarno geometrijo in se obenem znebimo napačnih ujemanj. Podrobnejši opis sledi v podpoglavju 2.6.

2.5 Ocena lege kamere

Ocenjevanje pozicije in orientacije (oz. lege) kamere delimo na relativno in absolutno. Razlika med njima je naslednja. Pri oceni relativne lege potrebujemo dve sliki. Za ta par slik poiščemo ujemanja značilnic (2D-2D korespondence) in na podlagi najmanj petih korespondenc izračunamo osnovno matriko E. Iz nje lahko neposredno pridobimo orientacijo in smer translacije druge kamere glede na prvo (za izhodišče koordinatnega sistema predpostavljamo, da je v prvi kameri). Takšno oceno lege uporabimo pri inicializaciji rekonstrukcije. V tem primeru magnituda translacije ni znana, zato pravimo da je rekonstrukcija določena do velikosti natančno. Osnovna matrika se med rekonstrukcijo uporablja tudi za filtriranje ujemanj.

Za naslednje poglede, ki so dodani k obstoječi rekonstrukciji, lahko določimo absolutno lego. Ta deluje na podlagi ujemanj med značilnicami posamezne slike in obstoječimi 3D točkami v rekonstrukciji (2D-3D korespondence). V tem primeru potrebujemo najmanj tri korespondence. Pozicija in orientacija nove kamere sta na ta način določena absolutno glede na lego prve kamere.

V obeh primerih se med korespondencami pojavijo napačna ujemanja, zato omenjena algoritma uporabljamo v kombinaciji z algoritmom RANSAC, ki lahko lego kamere določi tudi iz šumnih podatkov. Postopek je opisan v podpoglavju 2.6.

2.5.1 Ocena relativne lege kamere

Vsa informacija o relativni legi dveh kamer, je zajeta v pripadajoči osnovni matriki E. Naš cilj je izračunati to matriko na podlagi ujemanj značilnic. V našem delu smo uporabili obstoječo implementacijo algoritma predstavljenega v [31], ki za delovanje potrebuje pet ujemanj značilnic. V tem opisu bomo povzeli nekaj ključnih korakov, izpustili pa bomo nekatere matematične podrobnosti, ki so na voljo v originalnem delu.

Če spremenljivke epipolarne omejitve (2.6) pomnožimo med seboj, potem jo lahko zapišemo z enačbo (2.20). Ujemanje predstavljata par (p, p'), indeksi

spremenljivk pa so posamezni elementi vektorjev oz. matrik:

$$\begin{gathered}\hat{p}^{\mathsf{T}}\hat{E} = 0 \\ \hat{p} = [p_1p'_1, p_2p'_1, p_3p'_1, p_1p'_2, p_2p'_2, p_3p'_2, p_1p'_3, p_2p'_3, p_3p'_3]^{\mathsf{T}} \\ \hat{E} = [E_{11}, E_{12}, E_{13}, E_{21}, E_{22}, E_{23}, E_{31}, E_{32}, E_{33}]^{\mathsf{T}}.\end{gathered} \tag{2.20}$$

Iz petih ujemanj dobimo 5 epipolarnih omejitev, ki jih zberemo v matriki A dimenzije 5×9. Vsaka vrstica v matriki A predstavlja pripadajoči vektor $\hat{p}$. Z LU (ali SVD) razcepom pridobimo ničelni prostor te matrike, ki je 4-dimenzionalen. Rezultat so 4 linearno neodvisni vektorji. Njihova linearna kombinacija je osnovna matrika, ki jo iščemo (elemente 9-dimenzionalnih vektorjev preuredimo v 3×3 matrike). To linearno kombinacijo prikazuje enačba

$$E = xE_1 + yE_2 + zE_3 + wE_4. \tag{2.21}$$

Ker je E določena le do velikosti natančno lahko predpišemo $w = 1$, tako nam ostanejo še neznanke x, y in z. Njihove vrednosti poiščemo z upoštevanjem dodatnih omejitev, ki izvirajo iz lastnosti osnovne matrike. Z vstavitvijo enačbe (2.21) v omejitvi (2.22) in (2.23) pridobimo 10 polinomskih enačb tretje stopnje s tremi spremenljivkam:

$$\det(E) = 0 \tag{2.22}$$

$$2EE^{\mathsf{T}}E - \operatorname{tr}(EE^{\mathsf{T}})E = 0. \tag{2.23}$$

Prva enačba prispeva eno, druga pa preostalih devet. Sistem polinomskih enačb zapišemo z enačbo (2.24), kjer je matrika B dimenzije 10×20 in vsebuje koeficiente vseh polinomov. Stolpci matrike sovpadajo z vektorjem X vseh možnih monomov tretje stopnje treh spremenljivk.

$$BX = 0 \tag{2.24}$$

$$X = \left[x^3, x^2y, x^2z, xy^2, xyz, xz^2, y^3, y^2z, yz^2, z^3, x^2, xy, xz, y^2, yz, z^2, x, y, z, 1\right]^{\mathsf{T}}$$

Rešitve lahko sedaj poiščemo s splošno metodo za reševanje sistema polinomskih enačb. Matematično ozadje potrebno za razumevanje metode je

preobsežno za ta opis. Naj samo povzamemo, da je ideja postopka vpeljava Gröbnerjeve baze in t. i. akcijske matrike, katere lastni vektorji so rešitve sistema. Uporaba te metode na konkretnih problemih iz računalniškega vida je v celoti predstavljena v [32]. V našem primeru na podlagi sistema enačb (2.24) pridobimo 10 možnih rešitev za neznanke x, y in z, ki jih vstavimo nazaj v enačbo (2.21). Vse možne rešitve preizkusimo znotraj iteracije algoritma RANSAC in izberemo najboljšo.

Relativno lego dveh kamer pridobimo iz osnovne matrike E. Če je njen SVD razcep enak $E = U\ \mathrm{diag}(1,1,0)\ V^{\mathsf{T}}$, potem je projekcijska matrika druge kamere ena izmed rešitev v enačbi (2.25). Za prvo kamero privzamemo, da ima projekcijsko matriko $C = [I \mid 0]$. Vektor u_3 predstavlja tretji stolpec matrike U. Dokaz enačbe je na voljo v knjigi [33].

$$C' = \left[UWV^{\mathsf{T}}|u_3\right] \text{ ali } \left[UWV^{\mathsf{T}}|-u_3\right] \text{ ali } \left[UW^{\mathsf{T}}V^{\mathsf{T}}|u_3\right] \text{ ali } \left[UW^{\mathsf{T}}V^{\mathsf{T}}|-u_3\right]$$
$$\text{kjer je } \quad W = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.25}$$

Pravilno rešitev izberemo s triangulacijo ene izmed ujemajočih značilnic. Rekonstruirana točka namreč leži pred obema kamerama pri samo eni izmed štirih rešitev.

2.5.2 Ocena absolutne lege kamere

Omenili smo že, da moramo pri dodajanju slike k obstoječi rekonstrukciji oceniti lego nove kamere. To storimo na podlagi 2D-3D korespondenc (v nadaljevanju poglavja korespondence). V literaturi je splošen problem označen s kratico PnP (angl. Perspective-n-Point). V našem delu smo uporabili obstoječo implementacijo algoritma predstavljenega v [34], ki proizvede rešitev iz $n = 3$ korespondenc (P3P). To je tudi najmanjše število korespondenc za katero dobimo končno število rešitev. Prednosti tega algoritma v primerjavi z alternativnimi sta računska učinkovitost in numerična stabilnost.

Tu bomo opisali nekaj ključnih korakov omenjenega algoritma. Glavna ideja avtorjev je vpeljava dveh vmesnih koordinatnih sistemov, transformacijo med njima pa parametrizirajo samo z dvema parametroma. Za začetek definiramo nekaj spremenljivk. Pozicijo t in orientacijo R kamere pridobimo na podlagi treh 3D točk M_1, M_2 in M_3, ki ne smejo biti kolinearne. Za notranje parametre kamere predpostavljamo, da so poznani, zato lahko projekcije 3D točk predstavimo z vektorji f_1, f_2 in f_3, ki so iz izhodišča koordinatnega sistema kamere ν usmerjeni proti točkam na sliki. Nato iz f_1 in f_2 z enačbo (2.26) definiramo vmesni koordinatni sistem kamere $\alpha = (t, a_x, a_y, a_z)$:

$$\begin{aligned} a_x &= f_1 \\ a_z &= \frac{f_1 \times f_2}{\|f_1 \times f_2\|} \\ a_y &= a_z \times a_x \ . \end{aligned} \tag{2.26}$$

Tako lahko vektorje f_i z matriko $A = [a_x, a_y, a_z]^\mathsf{T}$ transformiramo v α. Če nam uspe določiti globalno orientacijo koordinatnega sistema α, potem lahko preko A določimo tudi orientacijo sistema ν (pozicija ostaja enaka t).

Iz točk M_1, M_2 in M_3 z enačbo (2.27) definiramo vmesni globalni koordinatni sistem $\beta = (M_1, b_x, b_y, b_z)$:

$$\begin{aligned} b_x &= \frac{\vec{M_1 M_2}}{\|\vec{M_1 M_2}\|} \\ b_z &= \frac{b_x \times \vec{M_1 M_3}}{\|b_x \times \vec{M_1 M_3}\|} \\ b_y &= b_z \times b_x \ . \end{aligned} \tag{2.27}$$

Podobno kot prej definiramo matriko $B = [b_x, b_y, b_z]^\mathsf{T}$, s katero 3D točke transformiramo v β. Tukaj je pomembna opazka, da lahko z določitvijo transformacije med β in α določimo tudi globalne koordinate kamere. Parametrizirati jo je možno s samo dvema parametroma ϕ in θ, kar je tudi razlog za vpeljavo vmesnih koordinatnih sistemov. Z enačbo (2.28) izračunamo pozicijo kamere v koordinatnem sistemu β, transformacijsko matriko iz β v α

pa z enačbo (2.29). V enačbi (2.28) je d_{12} razdalja med točkama M_1 in M_2. Spremenljivka $c = \cot\gamma$, kjer je γ kot med f_1 in f_2.

$$t^\beta(\phi,\theta) = \begin{bmatrix} d_{12}\cos\phi(\sin\phi\cdot c + \cos\phi) \\ d_{12}\sin\phi\cos\theta(\sin\phi\cdot c + \cos\phi) \\ d_{12}\sin\phi\sin\theta(\sin\phi\cdot c + \cos\phi) \end{bmatrix} \tag{2.28}$$

$$Q(\phi,\theta) = \begin{bmatrix} -\cos\phi & -\sin\phi\cos\theta & -\sin\phi\sin\theta \\ \sin\phi & -\cos\phi\cos\theta & -\cos\phi\sin\theta \\ 0 & -\sin\theta & \cos\theta \end{bmatrix} \tag{2.29}$$

Vrednosti parametrov ϕ in θ poiščemo s pomočjo transformacije točke M_3^β v α in upoštevanjem, da je njena smer enaka vektorju f_3^α. Tako pridemo do polinoma četrte stopnje iz katerega lahko analitično pridobimo vse štiri pare možnih rešitev (ϕ,θ). Celotna izpeljava enačb in polinoma, s podrobnejšim opisom, je podana v originalnem članku [34]. Za izbrano rešitev z enačbo (2.30) pridobimo lego kamere v globalnem koordinatnem sistemu:

$$\begin{aligned} t &= M_1 + B^\mathsf{T}\, t^\beta \\ R &= B^\mathsf{T}\, Q^\mathsf{T}\, A. \end{aligned} \tag{2.30}$$

Postopek torej proizvede štiri rešitve, med katerimi je ena pravilna. Poiščemo jo tako, da projekcijo nove 3D točke M_4 primerjamo z njeno dejansko lokacijo na sliki in izberemo rešitev kjer je napaka najmanjša.

2.6 Robustna ocena parametrov modela

V računalniškem vidu pogosto ocenjujemo parametre modela iz šumnih podatkov. V takšnih primerih si lahko pomagamo z algoritmom RANSAC (Random Sample Consensus) [25], ki omogoča robustno ocenjevanje parametrov. Gre za iterativni postopek, ki temelji na naključnem vzorčenju podatkov. Algoritem predpostavlja, da so vhodni podatki sestavljeni iz množice ki podpira model (angl. inliers) in množice osamelcev (angl. outliers), ki nimajo prispevka pri oceni modela.

Najprej bomo opisali splošno delovanje, nato pa konkretno uporabo v našem delu. Z algoritmom ocenjujemo parametre Θ modela $\mathcal{M}$ na vhodnih podatkih $\mathcal{D}$. Ena iteracija algoritma poteka na naslednji način. Najprej z naključnim vzorčenjem podatkov pridelamo podmnožico velikosti k označeno z $\mathcal{D}^k$. Število k je najmanjše število vzorcev, ki jih potrebujemo, da lahko z izbranim algoritmom $f_{\mathcal{M}}$ na množici $\mathcal{D}^k$ ocenimo parametre Θ in tako postavimo hipotezo $\mathcal{M}_\theta$. Podatke $\mathcal{D}$ nato razdelimo na dva dela, in sicer množico $\mathcal{I}$, ki podpira trenutno hipotezo, ter množico $\mathcal{O}$ osamelcev. Za to delitev mora biti definirana tudi funkcija napake $\xi_{\mathcal{M}_\theta}$, ki pri trenutni hipotezi za posamezni podatek izračuna odstopanje od predvidene vrednosti. Podatke smatramo za osamelce, če njihova napaka presega mejno vrednost t_ξ. Ena iteracija zelo malo verjetno pridela dobre parametre modela, zato postopek ponovimo n krat in na koncu obdržimo tisto hipotezo, za katero je vsota napak $\xi_{\mathcal{M}_\theta}$ (za podatke znotraj $\mathcal{I}$) najmanjša. Postopek je povzet v algoritmu 2.

Število iteracij ki jih potrebujemo, lahko ocenimo na naslednji način. Naj bo r_o delež osamelcev v podatkih. Potem lahko verjetnost, da algoritem najde dober model, zapišemo z enačbo

$$P_s = 1 - (1 - (1 - r_o)^k)^n. \tag{2.31}$$

Dober model je tisti, ki ni bil ocenjen z osamelci med podatki. To pomeni, da v vsaj eni izmed n iteracij pri vzorčenju k podatkov, ni bilo osamelcev. Če enačbo obrnemo, dobimo število potrebnih iteracij. V našem primeru za P_s uporabimo vrednost 0,99, parameter r_o pa ocenjujemo med izvajanjem algoritma.

$$n = \frac{\log(1 - P_s)}{\log(1 - (1 - r_o)^k)} \tag{2.32}$$

Prva uporaba tega algoritma v našem delu je pri ocenjevanju osnovne matrike E (model $\mathcal{M}$) in filtriranju ujemanj. Podatki $\mathcal{D}$ so vsa ujemanja med parom slik. Parametri Θ, so elementi osnovne matrike. Ocenjujemo jih z algoritmom opisanim v podpoglavju 2.5.1 (funkcija $f_{\mathcal{M}}$), ki za oceno potrebuje $k = 5$ vzorcev. Napaka predstavlja odstopanje značilnice od pri-

Algoritem 2 RANSAC

Vhod: Podatki $\mathcal{D}$, funkciji $f_{\mathcal{M}}$ in $\xi_{\mathcal{M}}$ ter parametra t_ξ in P_s

Izhod: Parametri Θ in množica podatkov $\mathcal{I}$, ki model podpira

1: **for** $i \leftarrow 1 \dots n$ **do**
2: $\quad \mathcal{D}_i^k \leftarrow$ Podmnožica podatkov velikosti k (enakomerno vzorčenje)
3: $\quad \Theta_i \leftarrow$ Parametri hipoteze $\mathcal{M}_\theta$ ocenjeni iz $\mathcal{D}_i^k$ z algoritmom $f_{\mathcal{M}}$
4: $\quad \mathcal{I}_i \leftarrow$ Podatki ki podpirajo hipotezo $\xi_{\mathcal{M}_\theta}(\mathcal{D}) \leq t_\xi$
5: $\quad \mathcal{O}_i \leftarrow$ Podatki ki ne podpirajo hipoteze $\xi_{\mathcal{M}_\theta}(\mathcal{D}) > t_\xi$
6: **end for**
7: $\Theta \leftarrow$ Najboljši parametri izmed $\Theta_1, \dots, \Theta_n$
8: $\mathcal{I} \leftarrow$ Podatki ki podpirajo najboljšo hipotezo

padajoče epipolarne premice. Meri se s Sampsonovo razdaljo [33] (funkcija $\xi_{\mathcal{M}}$). Filtrirana ujemanja predstavlja množica $\mathcal{I}$, ki podpira ocenjeni model.

Druga uporaba algoritma je pri oceni absolutne lege kamere iz 2D-3D korespondenc. Kamera, ki jo modeliramo, ima za parametre Θ translacijski vektor in rotacijsko matriko. Pridobimo jih z algoritmom opisanim v podpoglavju 2.5.2 (funkcija $f_{\mathcal{M}}$), ki za oceno potrebuje $k = 3$ vzorce. Kvaliteto ocene preverimo z reprojekcijsko napako (funkcija $\xi_{\mathcal{M}}$).

2.7 Triangulacija in minimizacija napake

Triangulacija je postopek, s katerim pridobimo pozicijo točke v 3D prostoru iz njenih pripadajočih projekcij na vsaj dveh različnih pogledih. Naj bo M iskana točka v 3D prostoru. Projekcijo točke M na eno izmed kamer s centrom c_i označimo s p_i. Za vse notranje in zunanje parametre kamere predvidevamo, da so poznani. Poltrak iz točke c_i skozi p_i definira vektor v_i oz. žarek, ki bi pri brezšumnih podatkih potekal tudi skozi točko M. V takšnem primeru, bi bil problem trivialen, saj bi preprosto poiskali presečišče dveh žarkov. Realni podatki neizogibno vsebujejo odstopanja pri meritvah, zato se žarki projekcij ne sekajo.

Obstaja več metod triangulacije, ki na različne načine definirajo optimalno lego točke M. V našem delu smo uporabili preprosto metodo srednje točke (angl. midpoint method) [35]. Kot namiguje že ime, ta metoda poišče sredino najkrajše daljice med žarkoma v_i in v_j različnih pogledov, ki pripadata isti točki M. Posplošena različica deluje za poljubno število žarkov, rešitev pa pridobimo z enačbo

$$M = \left[\sum_i (I - v_i v_i^{\mathsf{T}})\right]^{-1} \left[\sum_i (I - v_i v_i^{\mathsf{T}}) c_i\right]. \tag{2.33}$$

Gre za linearen sistem enačb, ki ga lahko rešimo zelo učinkovito. Njegova izpeljava je razložena v [35]. Za razliko od nekaterih alternativ [36], ta metoda ne minimizira reprojekcijske napake in zato ni najbolj natančna, njena prednost pa je časovna učinkovitost. Ocena lege je kljub temu dovolj dobra, saj napako minimiziramo v naslednjem koraku za celotno rekonstrukcijo.

Zadnji korak pri dodajanju slike k rekonstrukciji je minimizacija reprojekcijske napake za celotno množico slik (angl. bundle adjustment). Ta minimizacija izboljša natančnost rekonstrukcije in omili akumulacijo napake pri dodajanju slik. Reprojekcijska napaka je definirana kot seštevek razdalj med projekcijo posamezne 3D točke na vse slike na katerih je vidna in lokacijo pripadajočih značilnic na slikah. Naš cilj je poiskati parametre kamer in lege točk, ki minimizirajo to napako. Optimizacijski kriterij zapišemo z enačbo

$$\min_{M_i, P_j} \sum_{i=1}^{m} \sum_{j=1}^{n} v_{ij}\ d\big(P_j M_i\ ,\ p_{ij}\big)^2. \tag{2.34}$$

V zgornji enačbi je v_{ij} binarna spremenljivka, ki je postavljena na vrednost 1, če je 3D točka M_i vidna na kameri z indeksom j, sicer je postavljena na 0. Projekcijska matrika kamere je definirana s $P_j = KC_j$, kjer so K notranji in C zunanji parametri kamere. Detekcijo 3D točke M_i na kameri P_j predstavlja 2D točka p_{ij}, projekcijo te iste 3D točke na kamero pa zapišemo s $P_j M_i$. Napako med značilnico in projekcijo ocenimo s funkcijo razdalje d. Najbolj priljubljena izbira je Evklidska razdalja, uporabljene pa so lahko tudi druge mere [37].

Kamere P_j in točke M_i so parametri minimizacije za katere sicer imamo začetno oceno, vendar jih želimo izboljšati. Posamezna kamera ima v splošnem 11 parametrov (3 za pozicijo, 3 za orientacijo in 5 za notranje parametre), posamezna točka v prostoru pa 3. Skupno število parametrov optimizacije je torej $3m + 11n$. V našem primeru predpostavljamo, da so vse slike zajete z isto kamero za katero optimiziramo samo goriščno razdaljo. Preostali notranji parametri pri modernih kamerah ne prispevajo veliko k natančnosti, zato jih postavimo na privzete vrednosti. Število parametrov v našem primeru tako zmanjšamo na $3m + 6n + 1$. Dejanska optimizacija je opravljena z algoritmom Levenberg-Marquardt [38]. Gre za nelinearno metodo najmanjših kvadratov in je zelo pogosta izbira pri takšni vrsti optimizacije.

Minimizacija napake je časovno zahteven postopek, zato globalno minimizacijo izvedemo samo če se število točk v rekonstrukciji poveča za več kot 5 odstotkov. Po vsakem dodajanju slike pa izvedemo delno minimizacijo, ki optimizira 3D točke in poglede za zadnjih 20 slik.

Poglavje 3

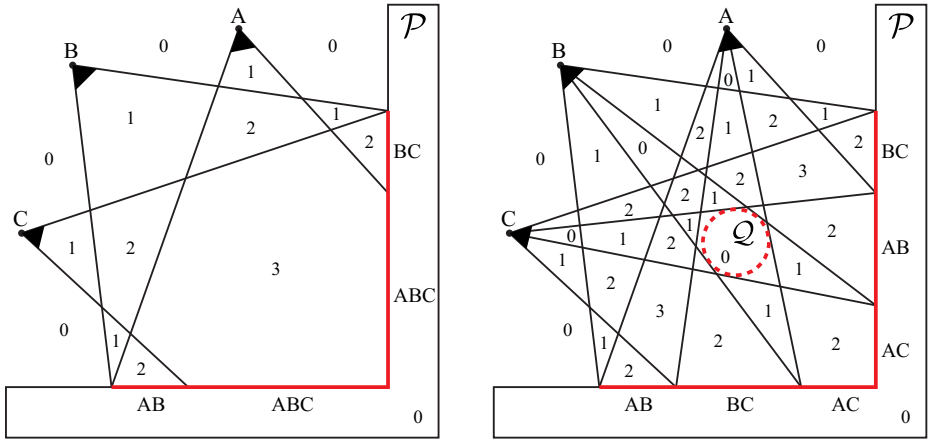
Gosta rekonstrukcija

3.1 Rekonstrukcija površine

Redka rekonstrukcija v določenih primerih uporabe ne zadostuje zaradi prenizke natančnosti, poleg tega pa ima predstavitev 3D modela z oblakom točk določene slabosti in posledično omejene možnosti uporabe. Za ceno dodatnega procesiranja, lahko model izboljšamo z algoritmi goste rekonstrukcije. V našem delu uporabljamo obstoječo implementacijo, ki sledi sodobnim pristopom in deluje v dveh korakih. Najprej je na podlagi oblaka točk rekonstruirana površina, nato pa je izboljšana ločljivost pridobljenega 3D modela.

Uporabljena implementacija za rekonstrukcijo površine sledi postopku predstavljenem v [39]. Gre za izboljšavo metode, ki je originalno predstavljena v [40]. Za razliko od klasičnih pristopov (npr. Poissonova rekonstrukcija površine [41]), je posebnost omenjenih metod to, da pri delovanju izkoriščajo tudi informacijo o legi kamer. Problem je formuliran z minimizacijo energijske funkcije na Delaunayevi tetraedralizaciji. Pridobljeni model je predstavljen s seznamom točk in trikotnikov oz. tako imenovano trikotniško mrežo (angl. triangular mesh). Takšna prestavitev je prostorsko učinkovita in dovolj fleksibilna za širok spekter uporabe. V našem delu je pridobljeni model uporabljen tudi kot izhodišče za oceno kvalitete in načrtovanje najboljšega naslednjega pogleda.

Osnova za delovanje metode je mera, ki jo avtorji poimenujejo podpora praznega prostora (angl. free space support). Ta mera poljubni točki v prostoru pripiše vrednost, ki predstavlja prepričanje v praznost prostora. Metoda upošteva vidnost vhodnih točk na kamerah, in zato omogoča rekonstrukcijo geometrije s slabo podporo oz. majhnim številom vhodnih točk, zgolj na podlagi prekrivanja predmetov. Skica na sliki 3.1 prikazuje opisani koncept.



Slika 3.1: Prikaz podpore praznega prostora. Površina predmetov se nahaja na prehodu iz ničelnih v pozitivne vrednosti. Skica je povzeta po [39].

Leva stran slike prikazuje tri kamere, ki opazujejo predmet $\mathcal{P}$. Z rdečo so obarvani segmenti, ki predstavljajo vhodne točke. Oznaka segmentov ponazarja njihovo vidnost na kamerah (npr. ABC pomeni da so točke segmenta vidne na kamerah A, B in C). V tem primeru je prostor implicitno razdeljen na območja. Vsakemu izmed njih je pripisana vrednost, ki predstavlja podporo praznega prostora. Na primer vrednost 3 pomeni, da skozi vsako točko tega območja potekajo trije segmenti, ki povezujejo eno izmed kamer z eno izmed vhodnih točk. Na desni strani slike v prostor vstavimo nov predmet $\mathcal{Q}$, ki nima vhodnih točk, vendar kljub temu zastira obstoječe točke. Kljub nespremenjeni množici vhodnih točk, se spremeni informacija o njihovi vidnosti. Ideja algoritma temelji na opazki, da se površina predmetov nahaja

na prehodu, kjer pride do velike spremembe vrednosti podpore praznega prostora (v idealnem primeru na prehodu iz ničelnih v pozitivne vrednosti). Opazka velja za dobro podprto ($\mathcal{P}$) in slabo podprto ($\mathcal{Q}$) geometrijo.

Konkreten postopek delovanja algoritma je naslednji. Najprej je prostor diskretiziran z Delaunayevo tetraedralizacijo [42]. Postopek je izvršen na uniji med vhodnimi točkami in množico centrov kamer. Vrednost podpore prostega prostora je izračunana za vsak tetraeder. Preden lahko podamo njeno točno definicijo moramo uvesti nekaj novih spremenljivk. Konstanta σ predstavlja najmanjšo razdaljo rekonstruirane površine. Izračunana je tako, da mediano dolžin vseh robov tetraedrov pomnožimo s faktorjem 2. Vhodne točke vsebujejo informacijo o vidnosti, zato jih zapišemo s parom $(c, p) \in S$, kjer je $c \in C$ center kamere, p vhodna 3D točka in S množica vseh parov. Podporo praznega prostora za tetraeder T izračunamo z enačbo

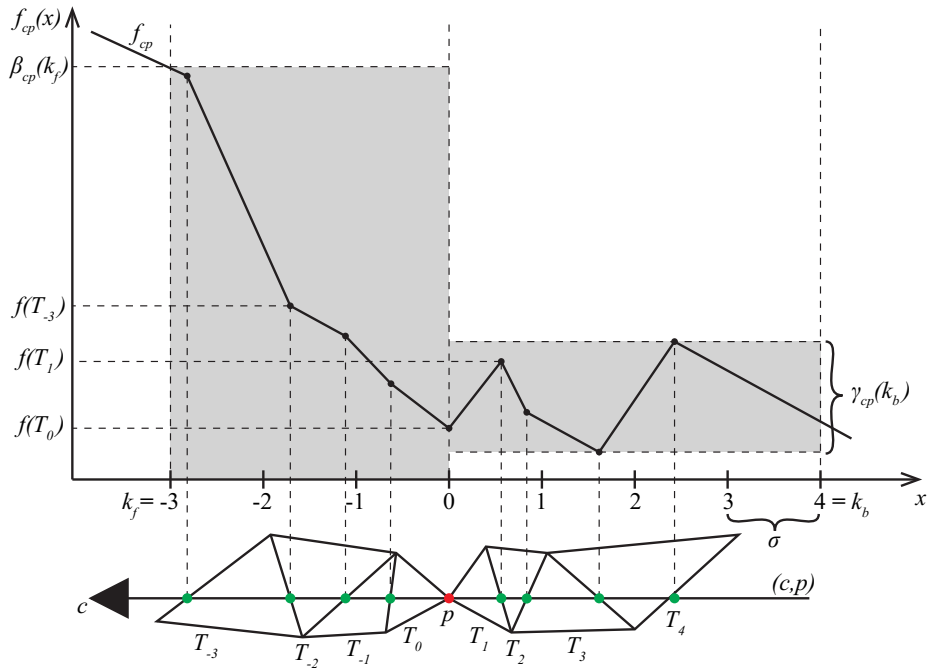
$$f(T) = \sum_{(c,p)\in S_T} \alpha(p); \qquad S_T = \{(c, p) \in S \mid (c, p) \cap T \neq \emptyset\}, \tag{3.1}$$

kjer je S_T množica segmentov (c, p), ki sekajo tetraeder T, $\alpha(p)$ pa je dodatna utež, ki predstavlja število točk v σ okolici točke p. Večja vrednost funkcije pomeni večje prepričanje, da je prostor ki ga zaseda tetraeder prazen. Na premici skozi točki c in p lahko sedaj definiramo novo funkcijo f_{cp}. Gre za odsekoma linearno funkcijo, ki interpolira vrednosti $f(T_i)$ za tetraedre, ki jih seka premica skozi (c, p). Njeno definicijsko območje je merjeno v enotah σ, točka p pa se nahaja na $\sigma = 0$. Pomožni funkciji, ki merita ekstremne vrednosti funkcije f_{cp} v okolici točke p definiramo z enačbama

$$\beta_{cp}(k) = \max_{x\in[-k,0]} \left(f_{cp}(x)\right), \quad \gamma_{cp}(k) = \frac{\max\limits_{x\in[0,k]} \left(f_{cp}(x)\right)}{2} + \frac{\min\limits_{x\in[0,k]} \left(f_{cp}(x)\right)}{2}. \tag{3.2}$$

Funkcija $\beta_{cp}(k)$ vrne maksimalno vrednost v smeri proti kameri, $\gamma_{cp}(k)$ pa povprečje maksimalne in minimalne vrednosti v smeri stran od kamere. Skica na sliki 3.2 prikazuje pomen funkcij. Definirajmo še dve pomožni funkciji za meritev absolutne in relativne spremembe funkcije f_{cp} v okolici točke p:

$$\epsilon^{\text{abs}}_{cp}(k_f, k_b) = \beta_{cp}(k_f) - \gamma_{cp}(k_b), \quad \epsilon^{\text{rel}}_{cp}(k_f, k_b) = \frac{\gamma_{cp}(k_b)}{\beta_{cp}(k_f)}. \tag{3.3}$$



Slika 3.2: Ilustracija funkcij f_{cp}, $\beta_{cp}(k)$ in $\gamma_{cp}(k)$ za premico skozi točki c in p. Skica je povzeta po [39].

Predhodne definicije so bile potrebne za realizacijo klasifikatorja, ki skuša za poljubno vhodno točko ugotoviti ali je ta res del površine:

$$K(c,p) = \begin{cases} 1 & \text{če } \left(\epsilon_{cp}^{\text{rel}}(k_f,k_b) < k_{\text{rel}}\right) \wedge \left(\epsilon_{cp}^{\text{rel}}(k_f,k_b) > k_{\text{abs}}\right) \wedge \left(\gamma_{cp}(k_b) < k_{\text{outl}}\right) \\ 0 & \text{drugače.} \end{cases} \tag{3.4}$$

Točko p klasificiramo kot del površine, če je relativna in absolutna sprememba funkcije f_{cp} v okolici točke p dovolj velika, količina osamelcev, ki jo ocenjuje funkcija $\gamma_{cp}(k)$, pa dovolj majhna. Uporabljene vrednosti parametrov so $k_f = 3$, $k_b = 4$, $k_{\text{rel}} = 1000$, $k_{\text{abs}} = 0{,}1$ in $k_{\text{outl}} = 400$. Več informacij o točnosti klasifikatorja in izbiri parametrov je na voljo v originalnem delu [39].

Površino modela poiščemo z reševanjem optimizacijskega problema na posebno pripravljenem s-t grafu. Vsak posamezni tetraeder predstavlja eno izmed vozlišč grafa $v \in V$, njegovi mejni trikotniki pa povezave s sosednjimi

tetraedri $e \in E$. Poleg tega uvedemo še dve posebni vozlišči, in sicer izvor s in ponor t, ki imata povezavo do vsakega izmed tetraedrov. Problem rešujemo z iskanjem minimalnega reza s-t grafa. Tako imenovani s-t rez grafa, vozlišča razdeli v disjunktni množici $\mathbb{S}$ in $\mathbb{T}$ (kjer velja $s \in \mathbb{S}$ in $t \in \mathbb{T}$), ki jim pripišemo oznako praznih in polnih tetraedrov. Cena reza je definirana kot seštevek uteži povezav, ki so odstranjene iz grafa. Cenovno oz. energijsko funkcijo reza zapišemo s spodnjo enačbo, kjer je w_{ij} utež na povezavi med vozliščema v_i in v_j, w_i^t in w_i^s pa sta uteži na povezavi vozlišča v_i s s in t:

$$c(\mathbb{S}, \mathbb{T}) = \sum_{\substack{v_i \in \mathbb{S} \setminus \{s\} \\ v_j \in \mathbb{T} \setminus \{t\}}} w_{ij} + \sum_{v_i \in \mathbb{S} \setminus \{s\}} w_i^t + \sum_{v_i \in \mathbb{T} \setminus \{t\}} w_i^s \ . \tag{3.5}$$

Rešitev problema je torej odvisna od uteži, ki jih postavimo na povezavah. Takšna formulacija problema nam omogoča, da časovno učinkovito poiščemo globalno optimalno rešitev. Gre za splošno metodo optimizacije, ki se uporablja tudi na drugih problemih računalniškega vida (npr. segmentacija). V našem primeru končno rešitev oz. površino modela predstavlja unija trikotnikov ki so na meji med polnim in praznim tetraedrom.

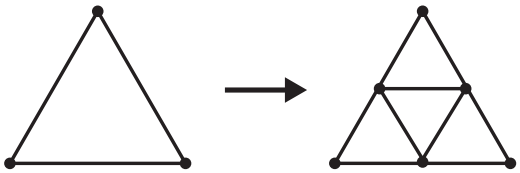
Podroben opis določanja uteži grafa je na voljo v [40] in [39], tu bomo povzeli le glavno idejo. Na začetku so vse uteži grafa postavljene na 0. Prispevek vhodnega para (c, p) na uteži je naslednji. Najprej za vrednost $\alpha(p)$ povečamo uteži w, vseh povezav ki pripadajo trikotnikom, ki jih seka segment (c, p). Za enako vrednost povečamo tudi utež w^s začetnega tetraedra, ki vsebuje točko c, in utež w^t končnega tetraedra, ki vsebuje točko p. Glavni prispevek dela [39] je dodatna okrepitev uteži w^t tistih tetraedrov, ki pripadajo točkam, ki so bile na podlagi (3.4) klasificirane kot del površine.

3.2 Izboljšava ločljivosti in natančnosti

Z dodatnim procesiranjem lahko začetnemu 3D modelu, ki ga pridobimo po rekonstrukciji površine, izboljšamo ločljivost in natančnost. Postopek uvrščamo na področje večpoglednega sterea. V preglednem članku [7] so

opisani tudi alternativni pristopi, v našem delu pa smo uporabili obstoječo implementacijo sodobne metode [43], ki je v našem primeru najbolj smiselna. Metoda deluje tako, da nad vozlišči modela definiramo t. i. energijsko funkcijo, natančnejšo pozicijo točk pa pridobimo z minimizacijo te funkcije.

Na začetku je ločljivost modela prilagojena ločljivosti vhodnih slik. Posamezen trikotnik je razdeljen na 4 manjše (slika 3.3), v primeru da njegova preslikava na vsaj enem paru kamer zaseda več kot 64 slikovnih elementov.



Slika 3.3: Skica delitve trikotnikov.

Nad vozlišči V tega gostejšega modela definiramo energijsko funkcijo, ki jo za posamezno vozlišče $v \in V$ zapišemo z enačbo

$$E(v) = E^P(v) + E^R(v). \tag{3.6}$$

Funkcija je sestavljena iz podatkovnega člena, ki temelji na fotometrični skladnosti (angl. photometric consistency), in regularizacijskega člena, ki je mera gladkosti geometrije. Natančnejšo pozicijo vozlišč določimo z minimizacijo te funkcije. Pri tem uporabimo metodo gradientnega spusta, kar pomeni da moramo izračunati njen gradient.

Fotometrična skladnost opisuje podobnost posameznih slikovnih elementov dveh slik. Zelo pogosta izbira je t. i. normalizirana prečna korelacija (angl. normalized cross correlation). Priljubljena je zaradi dobre odpornosti na spremembe pogleda in osvetlitve. Zapišemo jo z enačbo

$$M_{\mathrm{NCC}}(f,g) = \frac{(f-\mu(f))\cdot(g-\mu(g))}{\sigma(f)\sigma(g)}; \quad f = I_i(\Omega(p_i)),\ g = I_j(\Omega(p_j)). \tag{3.7}$$

Točki p_i in p_j sta slikovna elementa slik I_i in I_j. Njuno okolico pridobimo s funkcijo Ω, ki vrne sosednje slikovne elemente v oknu dimenzije 7×7. Velikost

okna je kompromis med enoličnostjo in invariantnostjo. Pridobljene elemente zapišemo z vektorjema f in g ter ju uporabimo pri izračunu podobnosti. Funkciji μ in σ vrneta povprečje in standardni odklon. Mera zasede vrednost na intervalu $[-1, 1]$, kjer večja vrednost pomeni večjo podobnost. Gradient prvega člena energijske funkcije je odvisen od fotometrične skladnosti. Vsak par slik I_i in I_j, na katerih je vidno vozlišče v, ima svoj prispevek pri izračunu, kar zapišemo z enačbo

$$\nabla E^P(v) = \sum_i \sum_{i \neq j} \nabla E^P_{ij}(v). \tag{3.8}$$

Člen $\nabla E^P_{ij}(v)$ je izračunan na naslednji način. Trikotniki, ki vsebujejo vozlišče v so preslikani nazaj na obe sliki. Člen $\nabla E^P_{ij}(v)$ je vsota prispevkov posameznih slikovnih elementov, ki so del preslikanih trikotnikov. Magnituda prispevka je odvisna od odvoda fotometrične skladnosti slikovnega elementa (glede na drugo sliko) in njegovih baricentričnih koordinat na trikotniku. Smer prispevka je odvisna od normale trikotnika. Izpeljava člena je preobsežna za ta opis, na voljo pa je v delu [44].

Regularizacijski člen $E^R(v)$ namesto eksplicitne definicije zapišemo kar neposredno v obliki gradienta. Najprej definiramo Laplaceov operator $\Delta(v)$, ki za podano vozlišče vrne vektor v smeri središča sosednjih vozlišč. Na podobne način definiramo še bi-Laplaceov operator, ki je ekvivalenten dvakratni uporabi Laplaceovega operatorja. Zapišemo ju z enačbama

$$\Delta(v) = \frac{\sum_{u_i \in \mathcal{N}} u_i}{|\mathcal{N}|} - v, \qquad \Delta^2(v) = \frac{\sum_{u_i \in \mathcal{N}} \Delta(u_i)}{|\mathcal{N}|} - \Delta(v). \tag{3.9}$$

Spremenljivka $\mathcal{N}$ predstavlja sosednja vozlišča vhoda v. Gradient $\nabla E^R(v)$ je definiran kot linearna kombinacija operatorjev $\Delta(v)$ in $\Delta^2(v)$. Gradient energijske funkcije lahko sedaj zapišemo s spodnjo enačbo, kjer sta α in β dodatni uteži linearne kombinacije:

$$\nabla E(v) = \sum_i \sum_{i \neq j} \nabla E^P_{ij}(v) + \alpha \Delta\left(v\right) - \beta \Delta^2\left(v\right). \tag{3.10}$$

3.3 Teksturiranje 3D modela

Teksturiranje je ključnega pomena za dosego realističnega izgleda rekonstruiranih modelov. Tekstura je dodatna slika, ki je priložena modelu, nad trikotniki pa je določena preslikava, ki slikovne elemente teksture preslika na njihovo površino. Teksturiranje ni trivialen problem zaradi sprememb v osvetlitvi, razlike v ločljivosti slik in zastiranja delov predmeta. V literaturi teksturiranju modelov ni namenjeno toliko pozornosti kot njihovi rekonstrukciji, kljub temu pa lahko zasledimo nekaj uspešnih metod. V našem delu uporabljamo obstoječo implementacijo sodobne metode [45].

Algoritem deluje v dveh korakih. V prvem koraku vsakemu trikotniku T_i pripišemo oznako l_i, ki določa en pogled, ki bo uporabljen za njegovo teksturiranje. Problem je ponovno formuliran z minimizacijo energijske funkcije. Zapišemo jo s spodnjo enačbo, kjer $\mathcal{N}$ predstavlja množico sosednjih trikotnikov posameznega trikotnika:

$$E(l) = \sum_{T_i \in T} E^P(T_i, l_i) + \sum_{(T_i, T_j) \in \mathcal{N}} E^S(T_i, T_j, l_i, l_j). \tag{3.11}$$

Optimizacija funkcije poteka z iskanjem minimalnih rezov na posebno pripravljenem grafu. Vozlišča grafa predstavljajo trikotniki, povezave pa njihovi robovi, ki mejijo na sosednje trikotnike. Graf vsebuje še dve dodatni terminalni vozlišči, ki imata povezave do vseh preostalih vozlišč. Postopek optimizacije je soroden metodi, ki je opisana v podpoglavju 3.1, le da je zaradi večjega števila možnih oznak potrebna nekoliko bolj kompleksna metoda. Predstavljena je v [46]. Gre za iterativno iskanje minimalnih rezov s t.i. α-ekspanzijo. V delu je opisano tudi, kako na podlagi funkcije (3.11) določimo uteži na povezavah omenjenega grafa.

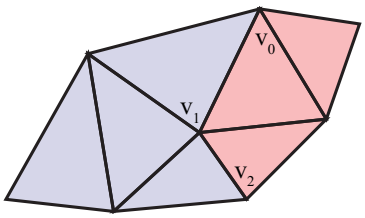
Prvi člen $E^P(T_i, l_i)$ funkcije (3.11) določa kako dober je pogled l_i za tekstruriranje trikotnika T_i. Izračunamo ga z enačbo

$$E^P(T_i, l_i) = - \sum_{\phi(F_i, l_i)} \|\nabla(I_{l_i}(p))\|_2. \tag{3.12}$$

Magnituda gradienta slike $\|\nabla(I_{l_i}(p))\|_2$ je izračunana s Sobelovim operatorjem [35]. Spremenljivka p eden izmed slikovnih elementov preslikave $\phi(F_i, l_i)$,

ki trikotnik T_i preslika na pogled z oznako l_i. Vrednost tega člena je velika pri veliki površini projekcije in ostri sliki, ker funkcijo minimiziramo pa je predznak negativen. S pomočjo drugega člena $E^S(T_i, T_j, l_i, l_j)$ energijske funkcije skušamo minimizirati število sosednjih trikotnikov z različno oznako. Izračunamo ga na preprost način, in sicer če sta oznaki l_i in l_j različni, potem zasede vrednost 1 v nasprotnem primeru pa vrednost 0. Kosi vhodnih slik, ki so določeni za teksturiranje trikotnikov, so združeni v eni sliki oz. teksturi.

Drugi korak algoritma je globalna uskladitev barv oz. optimizacija konsistentnosti teksture, tako da med trikotniki, ki so teksturirani z različnimi slikami ni vidnih prehodov. Pri opisu delovanja si bomo pomagali s sliko 3.4, ki prikazuje enostaven primer.



Slika 3.4: Primer preproste geometrije, kjer barva ponazarja, da so trikotniki teksturirani z različnima slikama.

Na sliki so modro in rdeče obarvani trikotniki teksturirani z različnima slikama. Na prehodu so vozlišča v_0, v_1 in v_2. Robove vzdolž te meje imenujemo šiv (angl. seam). Vsako vozlišče v_i, ki se nahaja na šivu podvojimo in iz njega ustvarimo dve novi vozlišči v_i^L in v_i^R. Vozliščem sedaj pripada enolična barva $f_v \in \mathbb{R}^3$ na podlagi pripadajoče slike. Za vsako vozlišče izračunamo popravek barve v obliki vektorja $g_v \in \mathbb{R}^3$, in sicer z minimizacijo spodnje enačbe, kjer so $\mathcal{S}$ vozlišča na šivu in $\mathcal{K}$ pari sosednjih vozlišč teksturirani z isto sliko:

$$\underset{g}{\operatorname{argmin}} \left(\sum_{v \in \mathcal{S}} \left(f_{v^L} + g_{v^R} - (f_{v^R} + g_{v^R})\right)^2 + \frac{1}{\lambda} \sum_{(v_i, v_j) \in \mathcal{K}} \left(g_{v_i} - g_{v_j}\right)^2 \right). \quad (3.13)$$

Prvi člen enačbe poskrbi, da sta si popravljeni barvi vozlišč vzdolž šiva čim

bolj podobni, drugi člen pa minimizira velikost popravka znotraj enega kosa teksture. Popravki za posamezne teksturne elemente (angl. texel) so interpolirani na podlagi popravkov vozlišč g_v in baricentričnih koordinat.

V tem opisu smo predstavili glavne ideje algoritma, izpustili pa smo nekaj podrobnosti, ki so tudi pomembne za robustno delovanje. Na primer v prvem koraku pri izbiri pogleda za teksturiranje trikotnika avtorji upoštevajo tudi fotometrično skladnost med kandidati. S tem preprečijo, da bi za teksturiranje bila uporabljena slika, na kateri je rekonstruirani predmet prekrit z objektom, ki ni rekonstruiran (npr. pešci in ostali premikajoči objekti, ki lahko na posameznih slikah zastirajo pogled). Še en takšen primer je izračun spremenljivk f_{v^L} in f_{v^R} v enačbi (3.13). Namesto barve posameznega slikovnega elementa, sta spremenljivki izračunani kot uteženo povprečje slikovnih elementov do sosednjih vozlišč (vzdolž šiva). Podrobnejši opis korakov je na voljo v originalnem delu [45].

Poglavje 4

Načrtovanje naslednjega pogleda

4.1 Ocena kvalitete 3D modela

Omenili smo že, da v našem delu za predstavitev 3D modelov uporabljamo trikotniško mrežo. To je seznam točk M_i in trikotnikov T_i. Tako kot v [2], kvaliteto 3D modela definiramo s funkcijo Q, ki vsakemu trikotniku $T_i = \{M_1, M_2, M_3\}$ priredi realno vrednost. Takšno preslikavo lahko zapišemo z enačbo

$$\begin{aligned} Q : \{\mathbb{R}^3, \mathbb{R}^3, \mathbb{R}^3\} &\to \mathbb{R} \\ T_i &\mapsto Q(T_i). \end{aligned} \tag{4.1}$$

Želimo si, da prirejena vrednost čim bolje predstavlja kvaliteto končnega modela, ki je merjena z natančnostjo in pokritostjo. Na ta način ima uporabnik že med rekonstrukcijo na voljo informacije o tem, kateri deli modela so bolj oz. manj natančni. Tako lahko sprejme informirano odločitev o postavitvi naslednje kamere in s tem optimizira proces zajemanja slik. V našem delu smo oceno kvalitete uporabili tudi kot osnovo za načrtovanje naslednjih pogledov. V delu [2] so avtorji predstavili mero GSD (angl. ground sampling distance), ki predstavlja maksimalno ločljivost preslikave posameznega trikotnika nazaj na sliko. V našem delu predlagamo podobno, vendar nekoliko

spremenjeno mero imenovano PPA (angl. pixels per area), ki se je v našem primeru izkazala za bolj robustno. Predlagano mero eksperimentalno ovrednotimo v poglavju 6.3 in jo primerjamo z GSD, zato bomo v tem poglavju opisali obe meri.

Mera GSD se izračuna na naslednji način. Posamezni trikotnik T_i preslikamo nazaj na kamere C_j. Ploščino trikotnika nato delimo z maksimalno ločljivostjo preslikave in rezultat korenimo. Mero zapišemo z enačbo

$$Q_{GSD}(T_i) = \min_{C_j} \sqrt{\frac{A(T_i)}{P(T_i, C_j)}}. \tag{4.2}$$

V zgornji enačbi, funkcija P vrne število slikovnih elementov, ki jih trikotnik T_i pokrije na sliki kamere C_j, funkcija A pa vrne ploščino trikotnika v 3D prostoru. Manjša vrednost funkcije Q_{GSD} predstavlja večjo kvaliteto modela. Zaradi majhnih in manj vidnih trikotnikov ta mera proizvede tudi nekatere zelo izstopajoče vrednosti. To je problematično pri vizualizaciji, zato odrežemo $\alpha = 10\%$ največjih vrednosti, tako kot v originalnem delu [2].

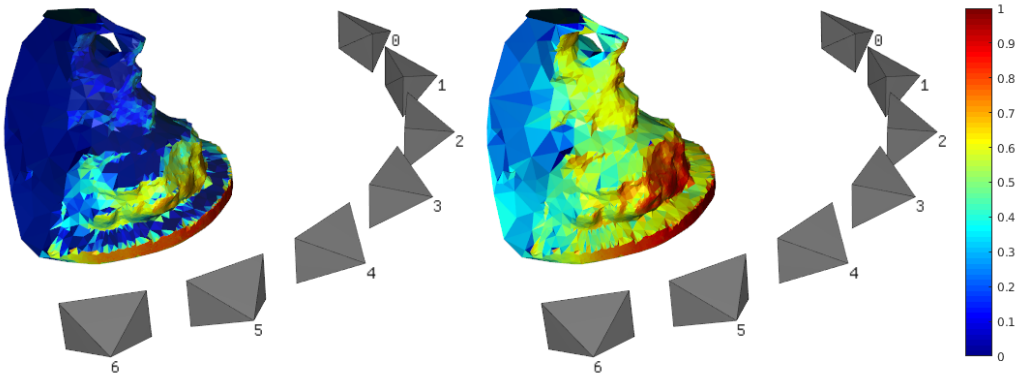
Mero PPA izračunamo na podoben način. Trikotnik T_i je ponovno preslikan nazaj na kamere C_j. Število slikovnih elementov preslikav tokrat seštejemo in delimo s ploščino trikotnika v 3D prostoru. Da omilimo vpliv večjih vrednosti, končni rezultat korenimo. Mero zapišemo z enačbo

$$Q_{PPA}(T_i) = \sqrt{\frac{\sum_{C_j} P(T_i, C_j)}{A(T_i)}}. \tag{4.3}$$

V tem primeru večja vrednost funkcije Q_{PPA} predstavlja večjo kvaliteto modela. Posebna normalizacija, kot pri meri GSD tukaj ni potrebna. V zgornjih enačbah je računsko najzahtevnejša funkcija P pri kateri moramo upoštevati medsebojno prekrivanje trikotnikov celotnega modela. Za učinkovito implementacijo uporabimo grafično kartico, tako da vsakemu trikotniku pripišemo unikatno število in upodobimo sliko na kamerah C_j. Rezultat preberemo iz pomnilnika in preštejemo slikovne elemente, ki pripadajo trikotniku. Na ta način funkcijo P izračunamo zelo učinkovito.

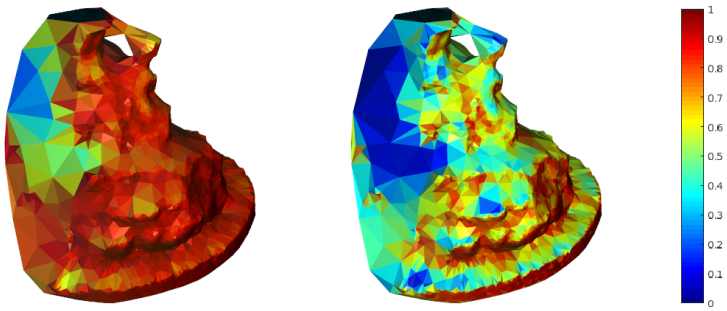
Na sliki 4.1 sta obe meri kvalitete vizualizirani na konkretnem primeru. Za primerjavo smo na sliki 4.2 prikazali tudi dejansko natančnost modela.

Postopek izračuna natančnosti je opisan v podpoglavju 6.1.3. Opazimo lahko, da se v tem primeru mera PPA vizualno bolje ujema z dejansko natančnostjo. V podpoglavju 6.3 z bolj obsežno evalvacijo pokažemo, da ima mera PPA boljši koeficient linearne korelacije z dejansko natančnostjo.



Slika 4.1: Vizualizacija kvalitete za konkreten 3D model. Na levi strani je prikazana mera GSD, na desni pa PPA. Meri sta normalizirani tako da zavzameta vrednosti na intervalu $[0, 1]$, kjer 1 predstavlja največjo in 0 najmanjšo kvaliteto. Opazimo lahko, da je bližjim trikotnikom in tistim z boljšo vidnostjo na kamerah, pripisana večja kvaliteta.

Obe meri predpostavljata, da boljša vidnost trikotnikov proizvede boljšo kvaliteto modela. Razlikujeta se v tem, da mera PPA upošteva vidnost trikotnika na vseh kamerah, medtem ko je pri meri GSD pomembna le čim večja ločljivost na eni izmed kamer. Omenjena predpostavka drži v primerih, ko ima rekonstruirani predmet dobro teksturo in je pridobljenih dovolj značilnic. V določenih primerih lahko model zaradi lukenj, odsevnih materialov, slabe teksture in šuma vsebuje trikotnike, ki se dejanski površini predmeta ne prilegajo, vendar so kljub temu dobro vidni na kamerah. V takšnih primerih opisani meri proizvedeta slabše rezultate.



Slika 4.2: Vizualizacija dejanske natančnosti modela za primerjavo z merama GSD in PPA. Leva stran prikazuje nespremenjeno natančnost. Za potrebe vizualizacije smo odstranili 10% največjih vrednosti. Na ta način omilimo vpliv velikih odstopanj in zmanjšamo razpon med vrednostmi. Rezultat je na desni strani. Rdeča barva predstavlja dobro natančnost, modra pa slabo.

4.2 Izbira najboljšega naslednjega pogleda

V splošnem je problem načrtovanja naslednjega pogleda (angl. next best view planning) definiran kot iskanje nove oz. dodatne postavitve senzorja z namenom izboljšave obstoječe rekonstrukcije oz. predstavitve prostora. Namesto barvne kamere so v splošnem lahko uporabljeni tudi drugi senzorji (npr. globinske kamere, laserski skenerji itd.). Odvisno od konteksta lahko na načrtovanje pogledov gledamo kot postopno gradnjo mreže kamer, strategijo za avtonomno raziskovanje, ali pa izbiro najboljših vhodnih slik pri rekonstrukciji iz velike množice podatkov. Tako kot v [15], v tem delu problem obravnavamo kot inkrementalni proces sistematične izboljšave natančnosti in pokritosti 3D modela.

V tem poglavju predstavimo nov pristop za načrtovanje naslednjega pogleda, ki temelji na oceni kvalitete 3D modela. Najprej definirajmo cenovno funkcijo, ki za podano lego kamere oceni kako dobra je njena postavitev.

Funkcijo zapišemo z enačbo

$$\begin{gathered} f_{NBV} : \mathbb{R}^{4\times4} \to \mathbb{R} \\ f_{NBV}(C) = \alpha * \mu\big(Q_{PPA}(v(C))\big) - \beta * \sigma\big(Q_{PPA}(v(C))\big). \end{gathered} \tag{4.4}$$

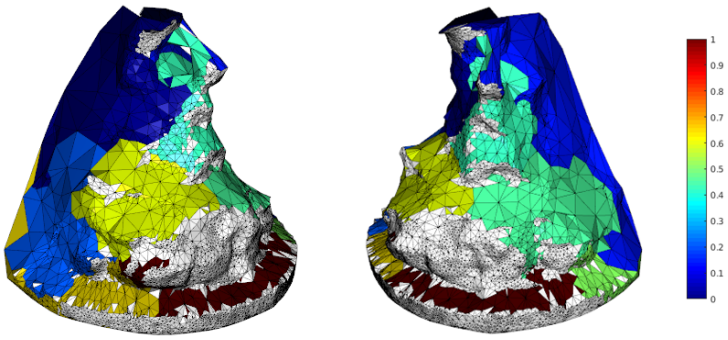
Spremenljivka C je 4×4 matrika lege kamere, ki vsebuje njeno pozicijo in orientacijo. Funkcija v za postavitev kamere C vrne vse vidne trikotnike, ki jih označimo s $\mathcal{T}$. Funkcijo Q, ki za podano množico $\mathcal{T}$ vrne pripadajoče kvalitete, smo definirali v prejšnjem podpoglavju. V našem primeru smo uporabili mero PPA. Funkciji μ in σ, za množico vrednosti $Q_{PPA}(\mathcal{T})$, vrneta njihovo povprečje in standardni odklon. Parametra α in β uravnavata njun prispevek pri izračunu. Ker gre za cenovno funkcijo iščemo takšno postavitev kamere, ki vrne čim manjšo vrednost. Utemeljitev takšne definicije je naslednja. Za novo kamero si želimo, da izboljša območja slabe kvalitete, po drugi strani pa je kamera lahko pravilno lokalizirana oz. dodana k rekonstrukciji, samo če so vidni tudi bolj kvalitetni deli modela. Želimo si torej čim večji razpon med vrednostmi. Ta kompromis med vidnostjo trikotnikov z dobro in slabo kvaliteto uravnavamo s parametroma α in β.

V delu [15], avtorji definirajo nekoliko bolj kompleksno cenovno funkcijo, ki sicer neposredno ne zajema informacije o kvaliteti modela, združuje pa informacije o negotovosti rekonstruiranih točk, velikosti površine modela preslikanega na kamero in o videzu teksture. Ker na tem področju standardna testna množica ne obstaja, je neposredna primerjava pristopov na žalost nemogoča. V poglavju 6.4 naš pristop primerjamo z referenčno postavitvijo kamer.

Hitrost izračuna funkcije f_{NBV} omejuje iskanje vidnih trikotnikov, pri katerem moramo upoštevati njihovo medsebojno prekrivanje. Kljub temu da je implementacija na grafični kartici sorazmerno hitra, je to še vedno najpočasnejši del izračuna, zato postopek načrtovanja naslednjega pogleda pričnemo z gručenjem trikotnikov modela, ki opravlja dve nalogi. Na podlagi gruč generiramo obvladljivo število postavitev kamere. Te postavitve predstavljajo kandidate za naslednji pogled. Pri velikih modelih je število gruč (in posledično kandidatov) lahko še vedno preveliko, zato so kandidati doda-

tno filtrirani na podlagi lokalne ocene za funkcijo f_{NBV}, ki je izračunana na podlagi trikotnikov v posamezni gruči. Na koncu funkcijo f_{NBV} izračunamo le za množico najboljših kandidatov, ki je dovolj majhna za hiter izračun.

Podrobnejši opis korakov je naslednji. Na začetku model naključno razdelimo na povezane gruče, ki vsebujejo med c_{min} in c_{max} trikotnikov. Normala posameznega trikotnika v gruči od povprečja ne sme odstopati za več kot ϕ stopinj. Primer gručenja je prikazan na sliki 4.3. Uporabljen je isti model in postavitev kamer kot na sliki 4.1. Za vsako gručo ovrednotimo lokalno oceno funkcije f_{NBV}, tako da namesto trikotnikov vidnih na kameri uporabimo trikotnike v gruči. Ker za lokalno oceno ne potrebujemo informacije o vidnosti, je ta izračun lahko zelo hiter. Gruče predstavljajo lokalno oceno cenovne funkcije, zato njihove vrednosti dodatno utežimo s številom trikotnikov. Pri večji velikosti je namreč lokalna ocena funkcije f_{NBV} bolj zanesljiva.



Slika 4.3: Primer gručenja na konkretnem primeru. Prikazana sta dva pogleda istega modela. Trikotniki iste gruče so obarvani z isto barvo. Barva predstavlja tudi lokalno oceno cenovne funkcije. Gruče na robu modela z nizko ceno (modra barva) predstavljajo dobre kandidate za postavitev kamere. Nekateri trikotniki niso del gruče (bela barva), saj njihova ocena kvalitete presega trenutno ciljno vrednost.

Za vsako gručo generiramo eno postavitev kamere, ki predstavlja kandidata za naslednji pogled. Kamera je postavljena od središča gruče G v smeri povprečja normal trikotnikov znotraj gruče. Usmerjena je proti centru gruče,

oddaljenost d pa je določena z enačbo

$$d = \gamma \sqrt[3]{A(G)}. \tag{4.5}$$

Funkcija A vrne ploščino gruče v 3D prostoru, korenska funkcija pa prepreči, da bi pri velikih gručah bila kamera preveč oddaljena. Dodaten parameter γ je odvisen od goriščne razdalje. V tem delu smo uporabili le dve različni kameri (virtualno in resnično), zato smo v obeh primerih ta parameter določili empirično. V podpoglavju 6.4 uporabljamo vrednost $\gamma = 4.0$, v podpoglavju 6.5 pa vrednost $\gamma = 3.0$. Z dodatnim testiranjem bi lahko formulirali tudi pravilo za določanje tega parametra. Za najboljših n kandidatov (v smislu lokalne ocene) nato izračunamo cenovno funkcijo f_{NBV} in za najboljši naslednji pogled predlagamo tistega z najmanjšo ceno. Opisano načrtovanje naslednjega pogleda je povzeto v algoritmu 3. Podane so tudi konkretne vrednosti parametrov, ki smo jih določili empirično.

Algoritem 3 Načrtovanje najboljšega naslednjega pogleda

Vhod: 3D model predstavljen s trikotniško mrežo M_T

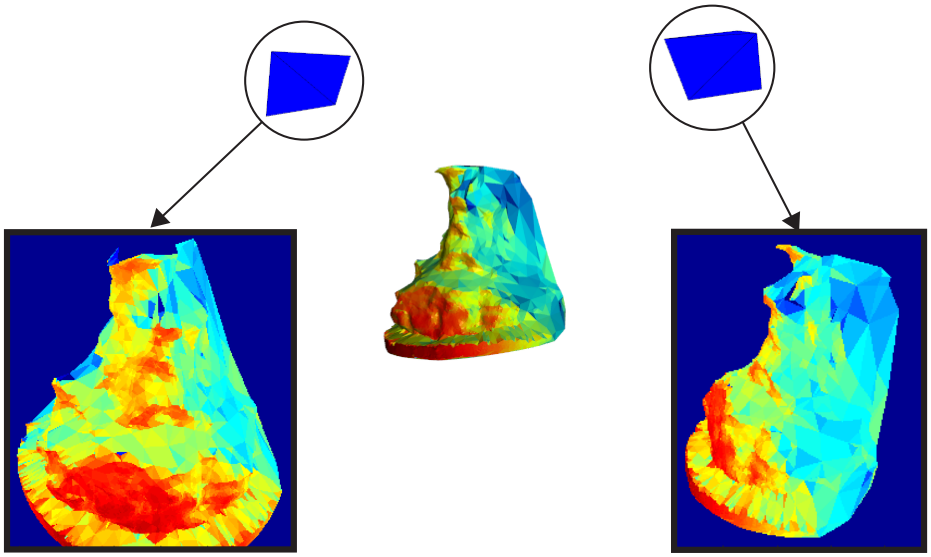
Izhod: Lega predlaganega pogleda C_{nbv}

1: $M_Q \leftarrow$ Izračun kvalitete za trikotniško mrežo $Q_{PPA}(M_T)$
2: $G_1 \dots G_n \leftarrow$ Delitev modela na gruče ($c_{min} = 100$, $c_{max} = 300$, $\phi = 100°$)
3: $f_{LOC}(G_i) \leftarrow$ Lokalna ocena funkcije f_{NBV} za vse posamezne gruče G_i na podlagi trikotnikov znotraj gruče ($\alpha = 1$, $\beta = 5$)
4: $C_1 \dots C_n \leftarrow$ Za vsako gručo G_i je generirana postavitev kamere C_i (kandidati za naslednji pogled)
5: $f_{NBV}(C_i) \leftarrow$ Ovrednotenje cenovne funkcije $f_{NBV}(C_i)$ za najboljših $n = 20$ kandidatov z najboljšo lokalno oceno $f_{LOC}(G_i)$ ($\alpha = 1$, $\beta = 5$)
6: $C_{nbv} \leftarrow$ Izbira najboljšega kandidata

Izboljšava kvalitete modela je postopna. Na začetku rekonstrukcije postavimo ciljno kvaliteto q_t. Ko q_p odstotkov trikotnikov doseže ciljno kvaliteto, parameter q_t povečamo za q_{inc}. Ti parametri so odvisni od uporabljene mere za kvaliteto. V našem primeru uporabljamo mero PPA in vrednosti parame-

trov $q_t = 1500$, $q_p = 85\%$ in $q_{inc} = 500$. Na ta način še dodatno spodbudimo raziskovanje in preprečimo morebitno kopičenje pogledov na enem območju.

Izračun cenovne funkcije f_{NBV} na konkretnem primeru je prikazan na sliki 4.4. Ponovno je uporabljen isti model in postavitev kamer kot na sliki 4.1, le da kamere rekonstrukcije tokrat niso prikazane. Na desni strani je prikazan najboljši predlagani pogled, na levi pa šesti najboljši. Prikazana je tudi upodobitev modela z mero PPA na obeh kamerah. Vrednost cenovne funkcije f_{NBV} za levi pogled znaša 30,3, za desni pogled pa $-111{,}9$. Desni pogled ima nižjo ceno zaradi večjega razpona vrednosti mere PPA, kar je razvidno tudi iz upodobitve modela. Ta pogled je boljša izbira, saj z njim zajamemo večje število slabše ocenjenih trikotnikov in s tem spodbujamo širitev rekonstrukcije, obenem pa z vidnostjo dobro ocenjenih trikotnikov omogočimo njegovo uspešno lokalizacijo. Levi pogled je postavljen blizu obstoječih kamer in ima manjši prispevek k izboljšavi rekonstrukcije.



Slika 4.4: Upodobitev modela z mero PPA za dva predlagana pogleda. Vrednost cenovne funkcije f_{NBV} za levi pogled znaša 30,3, za desni primer pa $-111{,}9$. Desni pogled je bolje ocenjen zaradi manjšega povprečja in večjega standardnega odklona mere PPA na vidnih trikotnikih.

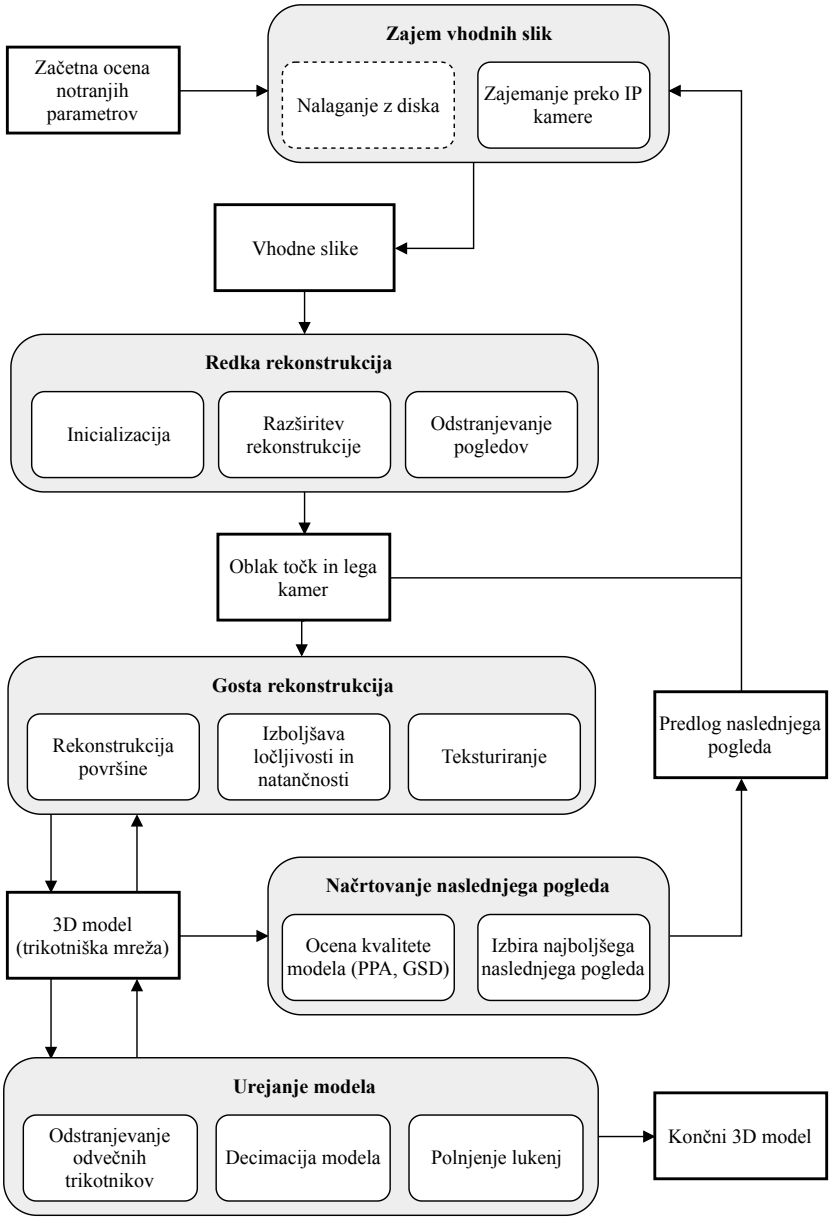
Poglavje 5

Implementacija rešitve

5.1 Predstavitev programske rešitve

V sklopu tega dela smo razvili programsko rešitev, ki vključuje algoritme opisane v prejšnjih poglavjih. Program podpira celoten proces rekonstrukcije, od zajemanja slik do tvorjenja končnega 3D modela s teksturo. V tem poglavju predstavimo njegove funkcionalnosti, knjižnice uporabljene pri implementaciji so podrobneje predstavljene v podpoglavju 5.2. Program je razdeljen na module za zajemanje slik, rekonstrukcijo, urejanje modela in načrtovanje pogledov. Za potrebe evalvacije smo pripravili še dodaten modul, ki omogoča rekonstrukcijo v simuliranem okolju. Moduli združujejo sorodne funkcionalnosti program.

Diagram na sliki 5.1 prikazuje odvisnost med funkcionalnostmi programa ter vhodnimi in izhodnimi podatki. Potek rekonstrukcije je naslednji. Slike so naložene z diska, ali pa so zajete preko IP kamere. Iz njih je postopoma grajena redka rekonstrukcija. Po vsaki dodani sliki je na podlagi oblaka točk rekonstruirana tudi površina. Tako pridobljeni 3D model je predstavljen s trikotniško mrežo. Na podlagi tega 3D modela, je izbran najboljši naslednji pogled, prikazana pa je lahko tudi ocena kvalitete. Ko je zajetih dovolj slik, lahko uporabnik iz modela odstrani odvečne trikotnike, ki so del okolice. Sledi izboljšava ločljivosti in teksturiranje, kar proizvede končni 3D model.



Slika 5.1: Oris programske rešitve. Beli okvirji z debelejšim robom predstavljajo vhode oz. izhode različnih funkcionalnosti programa, ki so prikazane v sivih okvirjih. Puščice prikazujejo potek rekonstrukcije in odvisnost med funkcionalnostmi.

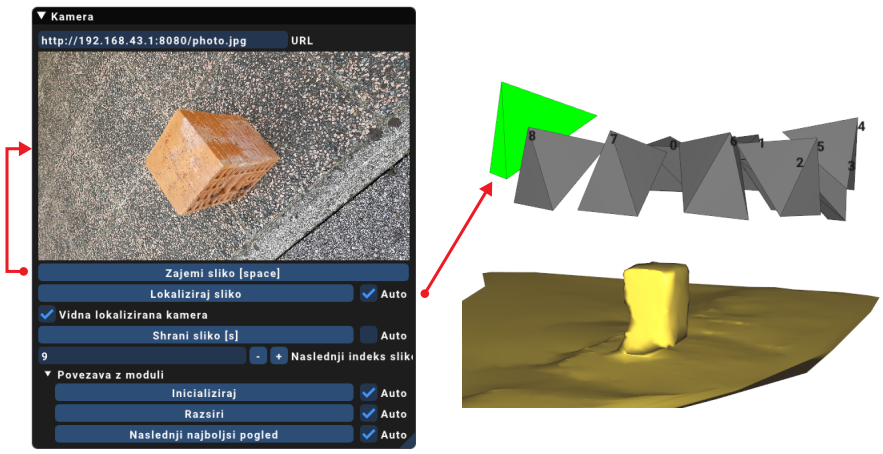
Na sliki 5.2 je prikazan zaslonski posnetek celotnega uporabniškega vmesnika. Z rdečo obrobo so označeni deli, ki predstavljajo funkcionalnosti posameznih modulov in so podrobneje opisani v nadaljevanju. Na sredini je primer 3D modela in postavitev kamer uporabljenih za rekonstrukcijo.

Slika 5.2: Zaslonski posnetek uporabniškega vmesnika.

5.1.1 Modul za zajemanje slik

V sklopu tega modula smo implementirali zajemanje slik preko IP kamere. Namen te funkcionalnosti je, da ima uporabnik interaktivno izkušnjo oz. sproten odziv o stanju rekonstrukcije za vsako zajeto sliko. Grafični vmesnik tega modula je prikazan na levi strani slike 5.3. V prvo polje vnesemo celoten naslov URL (angl. uniform resource locator) na katerega kamera objavlja slike. Ko pritisnemo gumb "Zajemi sliko", se iz podanega naslova prenese trenutna slika, ki jo vidi kamera. Z ukazom "Lokaliziraj sliko" lahko preverimo, če je zajeto sliko možno dodati k rekonstrukciji. V primeru, da je uspešno lokalizirana, se zeleno obarvana kamera postavi na ustrezno mesto (relativno

na obstoječo rekonstrukcijo), kot to prikazuje desna stran slike 5.3. V nasprotnem primeru je kamera obarvana rdeče, njena lega pa je neveljavna. Hitrost lokalizacije je v naši implementaciji odvisna predvsem od hitrosti prenosa slike in traja približno 1 s. S pohitritvijo prenosa (npr. z uporabo hitrejšega omrežja, paralelne implementacije ali USB povezave) in upoštevanjem lege predhodno lokalizirane slike, bi predvidoma dosegli približno 15 lokalizacij na sekundo, kar bi zadostovalo za izris uporabnikove kamere v realnem času. Če je zajeta slika ustrezna jo lahko shranimo s pritiskom na "Shrani sliko". Izbirno polje "Auto", tukaj in v nadaljevanju pomeni, da se ukaz izvede samodejno, ko je to mogoče. Na posnetku grafičnega vmesnika lahko vidimo še razdelek "Povezava z moduli". To so funkcionalnosti drugih modulov, ki bodo opisane v nadaljevanju. Tukaj so prisotne zato, da jih lahko izvedemo avtomatsko po shranjevanju slike. V primeru, da uporabnik funkcionalnosti tega modula ne potrebuje, so slike lahko naložene neposredno z diska. V podpoglavju 6.5 smo za rekonstrukcijo uporabili kamero mobilnega telefona, ki se lahko uporablja kot IP kamera (npr. z uporabo aplikacije IP Webcam [1])



Slika 5.3: Grafični vmesnik modula za zajemanje slik (levo) in prikaz lokalizacije kamere glede na obstoječo rekonstrukcijo (desno). Uspešno lokalizirana kamera je obarvana zeleno, obstoječe kamere rekonstrukcije pa sivo.

[1] Avtor Pavel Khlebovich. Na voljo za operacijski sistem Android.

Notranji parametri kamere so ob zagonu naloženi iz datoteke. Kot smo omenili v podpoglavju 2.1, je najpomembnejši med njimi goriščna razdalja. Ta parameter je med rekonstrukcijo optimiziran, zato zadostuje že groba ocena. Izračunamo ga lahko na primer iz širine vidnega polja θ_{FOV} in širine slikovnega senzorja w, s pomočjo enačbe

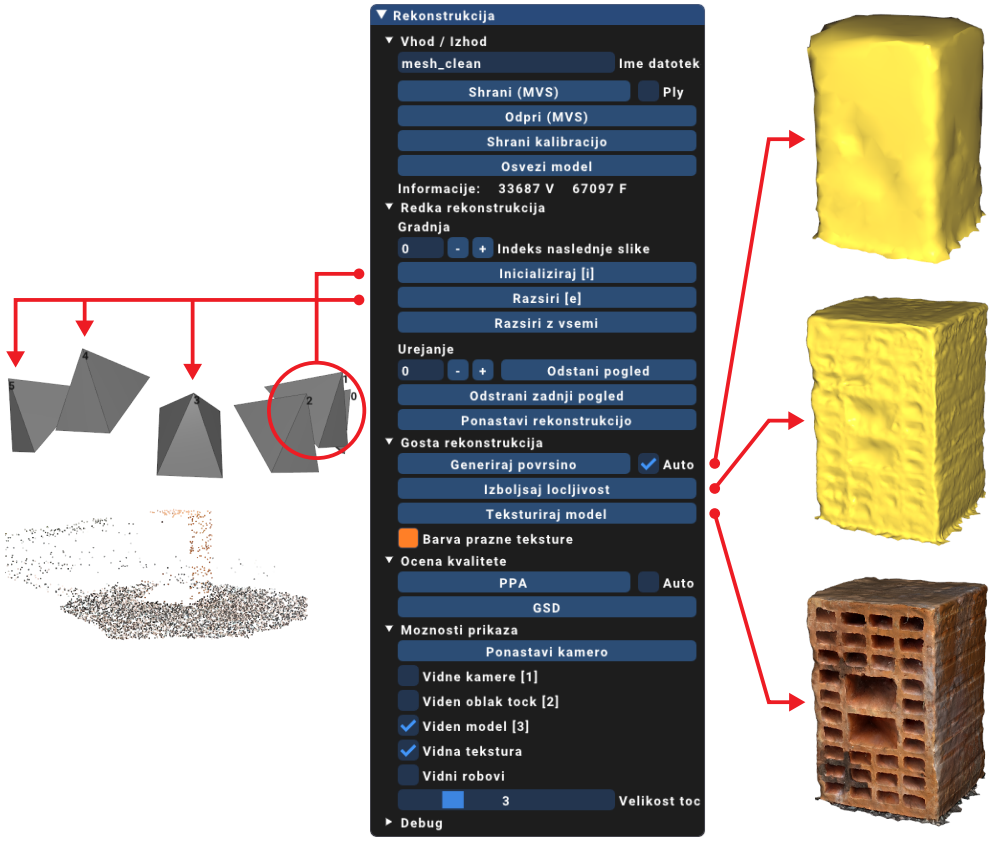
$$\theta_{FOV} = 2 \arctan\left(\frac{w}{f}\right). \tag{5.1}$$

Za bolj natančne rezultate je priporočljiva predhodna kalibracija kamere.

5.1.2 Modul za rekonstrukcijo

Ta modul zajema funkcionalnosti redke in goste rekonstrukcije. Njegov grafični vmesnik je prikazan na sredini slike 5.4. V razdelku "Vhod / Izhod" se nahajajo ukazi za shranjevanje in nalaganje modela. Rekonstrukcijo lahko inicializiramo, ko sta na voljo vsaj dve sliki. Naslednje slike dodamo k rekonstrukciji z ukazom "Razširi". Gradnjo redke rekonstrukcije z opisanima ukazoma prikazuje leva stran slike 5.4. Posamezne poglede lahko tudi odstranimo iz rekonstrukcije. Rekonstrukcija površine na podlagi oblaka točk proizvede model predstavljen s trikotniško mrežo. Ker gre za hitro operacijo, jo lahko izvedemo po vsaki dodani sliki. Izboljšava ločljivosti modela in tekstruriranje pa sta počasnejši operaciji. Preden ju izvedemo je zaželeno, da iz modela odstranimo odvečne trikotnike, ki so del ozadja.

Desna stran slike 5.4 prikazuje 3 modele. Zgornji model je rezultat redke rekonstrukcije, iz katerega so že bili odstranjeni odvečni trikotniki. Naslednji model je rezultat izboljšave ločljivosti, tretji model pa je končni tekstrurirani model. Modul ponuja še prikaz ocene kvalitete modela z ukazoma "PPA" in "GSD". Meri sta lahko prikazani, samo če je površina uspešno rekonstruirana.

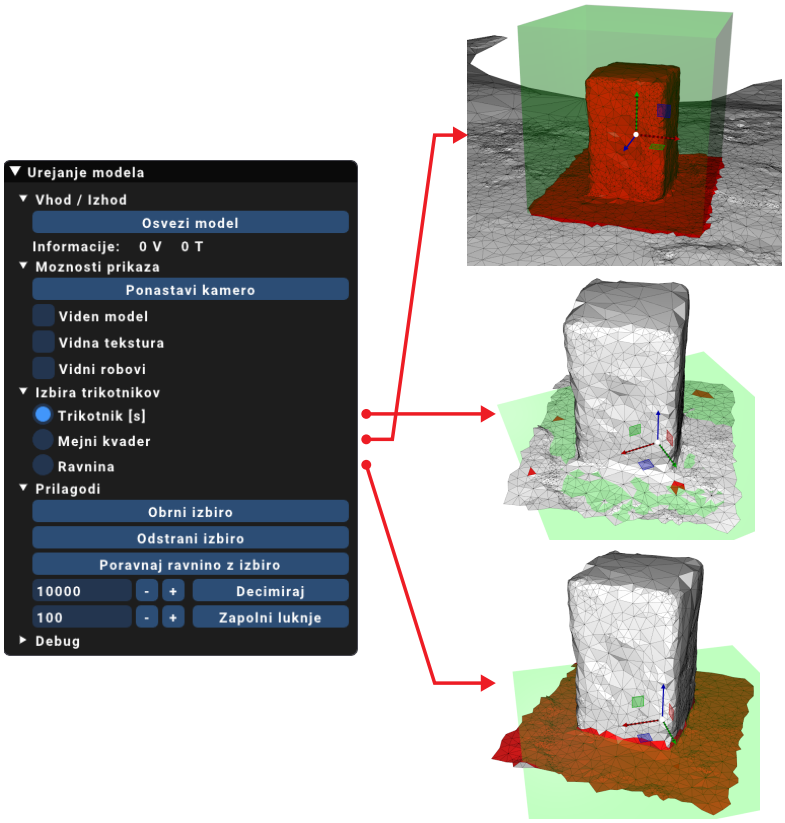


Slika 5.4: Grafični vmesnik modula za rekonstrukcijo (sredina), gradnja redke rekonstrukcije (levo) in vmesni koraki goste rekonstrukcije (desno).

5.1.3 Modul za urejanje

Glavni namen tega modula je odstranjevanje odvečnih trikotnikov in priprava modela na izboljšavo ločljivosti. Njegov grafični vmesnik je prikazan na levi strani slike 5.5. Na voljo so trije načini izbire trikotnikov in sicer izbira posameznih trikotnikov, izbira znotraj mejnega kvadra in izbira pod ravnino. Mejni kvader in ravnino lahko uporabnik interaktivno postavlja v prostoru. Primer uporabe vseh treh orodij je prikazan na desni strani slike 5.5. Najprej so izbrani trikotniki znotraj mejnega kvadra. Z ukazom "Obrni izbiro" izberemo komplement označenih trikotnikov in jih odstranimo z "Odstrani izbiro". Nato izberemo nekaj trikotnikov na ravni podlagi in kliknemo na

"Poravnaj ravnino z izbiro". Tako lahko izberemo trikotnike pod ravnino in jih odstranimo. Rezultat je očiščen model, ki je pripravljen za naslednje korake. Modul ponuja še funkciji za decimacijo modela in polnjenje manjših lukenj.

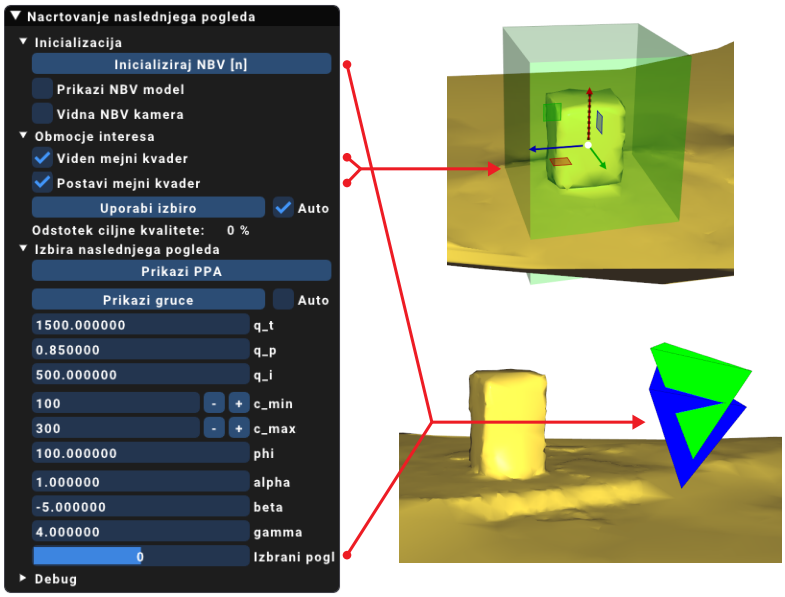


Slika 5.5: Grafični vmesnik modula za urejanje modela (levo) in načini izbire trikotnikov (desno).

5.1.4 Modul za načrtovanje pogledov

Za načrtovanje najboljšega naslednjega pogleda smo pripravili ločen modul. Njegov grafični vmesnik je prikazan na levi strani slike 5.6. Uporabnik ima možnost, da s pomočjo mejnega kvadra izbere območje interesa. To stori tako, da označi izbirni polji "Viden mejni kvader" in "Postavi mejni kvader"

ter potrdi izbiro z ukazom “Uporabi izbiro”. Na ta način algoritem upošteva le trikotnike znotraj mejnega kvadra oz. območja interesa. Z ukazom “Inicializiraj NBV” poženemo algoritem za načrtovanje naslednjega pogleda, ki vrne največ 20 predlaganih pogledov urejenih naraščajoče po vrednosti cenovne funkcije. V primeru, da najboljši predlagani pogled ne more biti lokaliziran, ali pa je fizično nedosegljiv, lahko z drsnikom izberemo naslednji pogled. Na predlagano mesto je postavljena modro obarvana kamera. Preostala vnosna polja služijo za nastavitve parametrov algoritma. Na desni strani slike 5.6 je prikazana postavitev mejnega kvadra (zgoraj) ter postavitev uporabnikove kamere (zelena) na predlagano mesto (modra). Pri postavitvi kamere na predlagano mesto v fizičnem prostoru so odstopanja neizogibna, kar je razvidno tudi na sliki.



Slika 5.6: Grafični vmesnik modula za načrtovanje naslednjega pogleda (levo), izbira območja interesa (desno zgoraj) in postavitev kamere na predlagano mesto (desno spodaj).

5.2 Uporabljene knjižnice

Programsko rešitev smo razvili v jeziku C++. Pri razvoju smo uporabili nekaj odprtokodnih knjižnic. Kot osnovo pri implementaciji redke rekonstrukcije smo uporabili knjižnico TheiaSfM [47]. Ta vsebuje številne algoritme s področja strukture iz gibanja med drugim tudi algoritme opisane v poglavjih od 2.3 do 2.7, z izjemo vizualnega slovarja. Postopnega dodajanja (ali odstranjevanja) slik k obstoječi rekonstrukciji v osnovi ne podpira, zato smo pri realizaciji postopka opisanega v poglavju 2.2 uporabili lastno implementacijo. Za hitrejše pridobivanje značilnic smo uporabili implementacijo algoritma SIFT na grafični kartici iz knjižnice SiftGPU [48]. Implementacijo vizualnega slovarja smo vzeli iz knjižnice COLMAP [18], ki je prav tako ena izmed priljubljenih knjižnic za rekonstrukcijo. Za gradnjo goste rekonstrukcije smo uporabili knjižnico OpenMVS [49], ki vsebuje algoritme za rekonstrukcijo površine iz oblaka točk, izboljšavo ločljivosti modela in teksturiranje. Za oceno kvalitete modela ter načrtovanje naslednjega pogleda smo uporabili lastno implementacijo. Omenjene komponente smo povezali v celoten sistem za gradnjo 3D modelov iz množice slik. Za interaktivno izkušnjo smo grafični vmesnik razvili s pomočjo knjižnice libigl [50]. Njen glavni namen je sicer procesiranje geometrije, vsebuje pa tudi preprost program za ogled 3D modelov, ki smo ga nekoliko prilagodili za naše potrebe. Elemente uporabniškega vmesnika (vnosna polja, gumbi itd.) priskrbi knjižnica Dear ImGui [51]. Za interaktivno postavitev mejnega kvadra in ravnine pa smo dodali še manipulator iz knjižnice ImGuizmo [52].

Poglavje 6

Eksperimentalna evalvacija

6.1 Metodologija evalvacije

Standardnega oz. uveljavljenega postopka za evalvacijo ocene kvalitete 3D modela in načrtovanja naslednjega pogleda v literaturi nismo zasledili. Problem je v tem, da bi v obeh primerih poleg fizičnega modela oz. njegovih posnetkov potrebovali tudi njegov referenčni 3D model (za izračun natančnosti). Pri načrtovanju pogledov pa je problematična tudi natančna postavitev kamere na predlagano mesto v fizičnem prostoru, kjer so odstopanja neizogibna. Iz teh razlogov smo v našem delu oceno kvalitete in načrtovanje naslednjega pogleda evalvirali z rekonstrukcijo računalniških 3D modelov. Rezultati so predstavljeni v podpoglavjih 6.3 in 6.4, nekaj primerov rekonstrukcij resničnih predmetov pa podamo v podpoglavju 6.5. Čeprav se na področju računalniškega vida želimo izogniti sintetičnim testnim zbirkam, je v tem primeru to najbolj praktična rešitev, ki odpira tudi možnosti za pripravo standardne testne množice za neposredno primerjavo različnih pristopov načrtovanja naslednjega pogleda. V podpoglavju 6.5 podamo še nekaj časovnih meritev pomembnejših korakov rekonstrukcije.

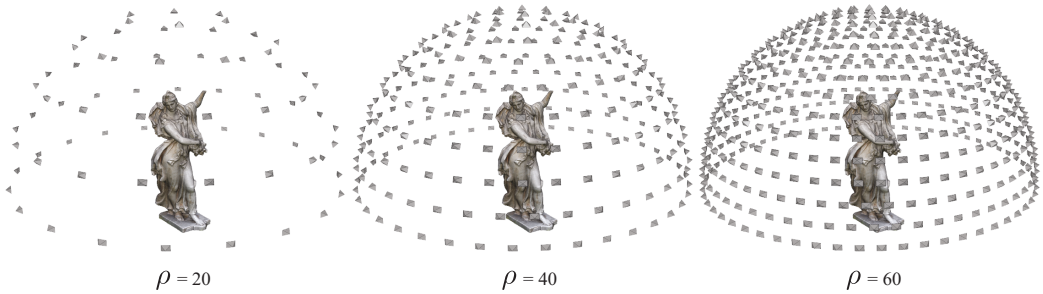
6.1.1 Enakomerna postavitev kamer

Pri evalvaciji smo uporabili referenčne 3D modele, ki so prosto dostopni na spletnem portalu Sketchfab [1] in so objavljeni pod licenco CC. Konkretni modeli uporabljeni za evalvacijo so podrobneje predstavljeni v podpoglavju 6.2.

V nadaljevanju bomo rezultate načrtovanja naslednjega pogleda primerjali s t. i. osnovno rekonstrukcijo. Ta je pridobljena iz enakomerne postavitve kamer. Kamere so razporejene enakomerno po navidezni polkrogli in so usmerjene proti njenemu središču. Njihova postavitev je generirana na naslednji način. Najprej izberemo gostoto postavitve ρ, ki predstavlja število kamer na prvem obroču imenovanem tudi glavni krogelni krog (presek krogle z ravnino, ki poteka skozi središče). Razmak med kamerami je določen z $d = 2\Pi/\rho$. Nov obroč kamer dobimo tako, da se pomaknemo za razdaljo d po polkrogli navzgor in postavimo kamere z medsebojnim razmakom d.

Primer enakomerne postavitve kamer je prikazan na sliki 6.1. V tem primeru so prikazane tri postavitve kamer, in sicer za gostote $\rho = \{20, 40, 60\}$, ki na vseh nivojih skupaj proizvedejo 72, 271 oz. 596 kamer (v tem vrstnem redu). Notranji parametri so znani, in jih med rekonstrukcijo ne optimiziramo. Osnovno rekonstrukcijo gradimo iz slik ločljivosti 2048×1536, ki jih upodobimo z virtualnimi kamerami. Najprej so dodane slike na spodnjem obroču, nato pa nadaljujemo z naslednjimi obroči proti vrhu. Primerjava z načrtovanjem naslednjih pogledov je podana v nadaljevanju. Za bolj pošteno primerjavo, smo referenčne modele izbrali tako, da je za njihovo rekonstrukcijo enakomerna postavitev kamer sorazmerno ugodna. V splošnem je postavitev kamer v obliki polkrogle zelo ugodna pri rekonstrukciji predmetov z večinoma konveksno obliko, ki jih lahko postavimo v središče polkrogle. Med drugim je takšna postavitev uporabljena tudi pri gradnji testnih množic za evalvacijo rekonstrukcijskih algoritmov [1, 53].

[1] https://sketchfab.com



Slika 6.1: Prikazana je enakomerna postavitev kamer za gostote $\rho = \{20, 40, 60\}$. Skupno število kamer z leve proti desni je 72, 271 in 596. V sredini polkrogel je eden izmed referenčnih modelov. Rezultat rekonstrukcije referenčnega modela s takšno postavitvijo kamer je t. i. osnovna rekonstrukcija, ki jo v nadaljevanju uporabimo pri evalvaciji.

6.1.2 Poravnava modelov

Preden lahko ovrednotimo natančnost in pokritost, moramo rešiti še problem poravnave modelov. Koordinatna sistema rekonstrukcije in referenčnega modela v splošnem namreč nista poravnana in sta različnih velikosti. Pogosto uporabljen algoritem za poravnavo oblakov točk imenovan ICP (angl. iterative closest point) [54], je v tem primeru manj primeren, saj je rekonstrukcija med gradnjo pomanjkljiva in ima drugačno gostoto točk od referenčnega modela. V našem primeru za bolj natančno poravnavo izkoristimo poznano lego virtualnih kamer s katerimi upodobimo slike referenčnega modela. Če predpostavimo, da je njihova rekonstruirana lega določena popolnoma natančno, potem lahko poravnavo med poznanimi in rekonstruiranimi kamerami opišemo z enolično linearno transformacijo, ki zajema rotacijo, translacijo in skaliranje. S poravnavo kamer posledično poravnamo tudi rekonstrukcijo in referenčni model. Kljub temu da v praksi lokalizacija kamer ni popolna, je v našem primeru njihova lega določena dovolj natančno, tako da je napaka pri poravnavi za red velikosti manjša od natančnosti rekonstrukcije in zato ne predstavlja posebnih težav. Primer poravnave rekonstruiranega modela z referenčnim je prikazan na sliki 6.2.

Slika 6.2: Prikaz poravnave delne rekonstrukcije z referenčnim 3D modelom. Na levi strani lahko vidimo, da trenutna rekonstrukcija (rumena geometrija) ni poravnana z referenčnim modelom, zato izračun natančnosti in pokritosti ni mogoč. Linearna transformacija med koordinatnima sistemoma modelov je izračunana na podlagi poznane lege virtualnih kamer in njihove rekonstruirane lege. Poravnana modela sta prikazana na desni strani.

Konkreten postopek s katerim pridobimo transformacijo je opisan v [55]. Kratek povzetek algoritma je naslednji. Pozicijo virtualnih kamer označimo z množico $A = \{a_1, \ldots, a_n\}$ in pozicijo njihove rekonstruirane lege z $B = \{b_1, \ldots, b_n\}$, kjer so $a_i, b_i \in \mathbb{R}^3$ (orientacija tukaj ni uporabljena). Točki $\{a_i, b_i\}$ predstavljata ujemajoči par za katerega želimo da je razdalja čim manjša. Iskano transformacijo predstavljajo rotacija $R \in \mathbb{R}^{3\times3}$, translacija $t \in \mathbb{R}^3$ in skaliranje $s \in \mathbb{R}$, kar lahko zapišemo z enačbo

$$B = sR\ A + t. \tag{6.1}$$

Iz podatkov najprej odstranimo translacijsko komponento in pripravimo kovariančno matriko H, za katero izračunamo razcep SVD. V spodnji enačbi sta $\bar{A}$ in $\bar{B}$ centroida oz. povprečji vhodnih točk:

$$\begin{aligned} H &= \sum_{i=1}^{n} (a_i - \bar{A})(b_i - \bar{B})^{\mathsf{T}} \\ [U, S, V] &= \mathrm{SVD}(H)\ . \end{aligned} \tag{6.2}$$

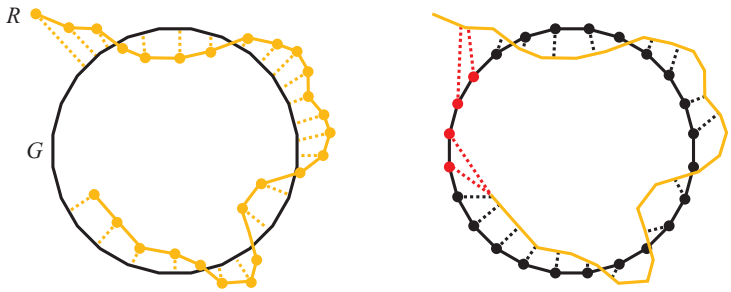
Z razcepom SVD pridobimo tri nove matrike, na podlagi katerih izračunamo

parametre poravnave, kot to prikazujejo enačbe

$$\begin{aligned} R &= VU^{\mathsf{T}} \\ t &= \bar{B} - R\,\bar{A} \\ s &= \left(\sum S\right) / \left(\sum \|a_i - \bar{A}\|_2\right). \end{aligned} \tag{6.3}$$

6.1.3 Izračun natančnosti in pokritosti modela

Za primerjavo rekonstrukcije z referenčnim modelom je v literaturi uveljavljen pristop prvotno predstavljen v delu [1]. Referenčni model označimo z G, rezultat rekonstrukcije pa z R. V delu smo že večkrat omenili da kvaliteto rekonstrukcije merimo z natančnostjo in pokritostjo modela. Tukaj predstavimo natančnejšo definicijo obeh pojmov. Meri sta izračunani na podlagi razdalje med R in G. Pomen te razdalje je grafično prikazan na sliki 6.3.



Slika 6.3: Leva stran prikazuje razdaljo od rekonstrukcije R do referenčnega modela G. Za vsako točko v R poiščemo najbližjo točko v G. Na podlagi teh razdalj je izračunana natančnost. Desna stran prikazuje izračun razdalje v drugo smer (od G do R) na podlagi katere izračunamo pokritost. Na skici je razdalja za rdeče točke prevelika, zato jih R ne pokrije. Skica povzeta po [1].

Pri izračunu natančnosti moramo za vsako točko v R poiskati najbližjo točko v G. Modeli so v našem delu predstavljeni v obliki trikotniške mreže. Pri takšni predstavitvi so velikosti trikotnikov (oz. ločljivost modela) lahko zelo neenakomerne. Pri računanju razdalje zato predhodno iz originalnega modela, z enakomernim vzorčenjem, pridelamo oblak točk, ki je uporabljen

za primerjavo. S tem izničimo učinek različnih velikosti trikotnikov. Mero natančnosti predstavlja tista razdalja A_d, za katero je 90% točk v R, do pripadajočih najbližjih točk v G, oddaljenih za največ A_d. To mero poimenujemo *nenatančnost* in si želimo, da je pri rekonstrukciji čim manjša.

Pri izračunu pokritosti iščemo razdaljo v drugi smeri, in sicer za točke v G poiščemo najbližje točke v R. Za posamezne točke referenčnega modela pravimo da so pokrite, če je njihova razdalja do rekonstrukcije dovolj majhna. Tukaj je potrebno uvesti še dodatno mejno vrednost C_t, ki določa kdaj je razdalja še sprejemljiva. Ker absolutna velikost referenčnih modelov ni znana, je to mejno vrednost potrebno določiti za vsak model posebej. Pri tem si pomagamo z algoritmom PCA (angl. principal component analysis). Gre za splošen algoritem, ki za vhodne podatke poišče novo ortogonalno bazo, tako da bazni vektorji zajemajo čim več variance v podatkih. V našem primeru je vhod referenčni model predstavljen z oblakom točk. Pri evalvaciji smo za mejno vrednost C_t izbrali 1% dolžine najdaljšega baznega vektorja. Mero pokritosti predstavlja delež točk G, za katere je razdalja do R manjša od C_t. Za hitro iskanje najbližjih sosedov smo uporabili implementacijo k-d drevesa iz knjižnice nanoflann [56].

6.2 Podatki za evalvacijo

Referenčni modeli [2] uporabljeni pri evalvaciji so prikazani na sliki 6.4. Z leve proti desni in od zgoraj navzdol si sledijo: štor (11352 vozlišč, 21876 trikotnikov), jabolko (15002 vozlišč, 30000 trikotnikov), kamen (17096 vozlišč, 34188 trikotnikov), krava (14469 vozlišč, 27975 trikotnikov), fontana (44843 vozlišč, 89129 trikotnikov), nagrobnik (25611 vozlišč, 49998 trikotnikov), kip (109472 vozlišč, 218214 trikotnikov), hidrant (46255 vozlišč, 90213 trikotnikov), storž (14964 vozlišč, 29992 trikotnikov). Ločljivost njihovih tekstur znaša 4096×4096 slikovnih elementov.

[2] Uporabniška imena avtorjev na portalu Sketchfab so: GSXNet (hidrant), jvouillon (nagrobnik), misterdevious (kip) in 3dhdscan (preostali modeli).

Slika 6.4: Referenčni modeli uporabljeni pri evalvaciji.

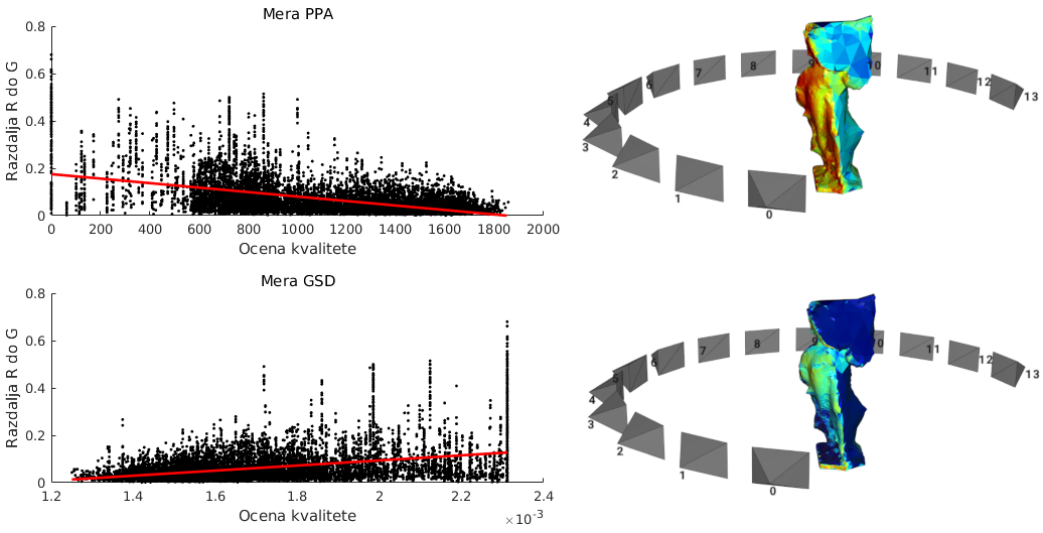
6.3 Evalvacija mere za kvaliteto

V tem poglavju predstavimo rezultate evalvacije nove mere za kvaliteto PPA, in jo primerjamo z obstoječo mero GSD. Za obe meri preverimo kako dobro odražata dejansko natančnost rekonstrukcije. To storimo z izračunom Pearsonovega koeficienta korelacije, ki je definiran z enačbo

$$r_{xy} = \frac{\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n} (x_i - \bar{x})^2} \sqrt{\sum_{i=1}^{n} (y_i - \bar{y})^2}}. \tag{6.4}$$

Gre za najpogosteje uporabljeno mero linearne povezanosti dveh številskih spremenljivk. Koeficient lahko zavzame vrednosti na intervalu $[-1, 1]$, kjer vrednosti ± 1 predstavljata popolno linearno odvisnost, vrednost 0 pa pomeni, da spremenljivki nista odvisni.

Pri evalvaciji smo uporabili 4 referenčne modele, in sicer fontano, nagrobnik, hidrant in kip. Za vsak model smo zgradili rekonstrukcijo z enakomerno postavitvijo kamer gostote $\rho = 20$ (skupno 72 kamer). Po vsaki dodani kameri, smo za vmesni model izračunali razdaljo do referenčnega modela in obe meri kvalitete. Pri izračunu razdalje smo modele predhodno enakomerno vzorčili. Za referenčni model smo uporabili 100000 točk, za rekonstruirani model pa 3-kratnik števila vozlišč. Na sliki 6.5 je prikazan primer korelacije med natančnostjo in oceno kvalitete za meri PPA in GSD. Primer prikazuje vmesni model rekonstrukcije, ki je zgrajen iz 14-ih pogledov. Na levih grafih je za vsako točko modela prikazana njena razdalja do reference (os y) in ocena kvalitete (os x). Na obeh grafih je razvidno, da je napaka večja pri točkah s slabše ocenjeno kvaliteto. V tem primeru je vrednost koeficienta korelacije za mero PPA $-0{,}52$, za mero GSD pa 0,44. Predznak se razlikuje, ker pri meri PPA večja vrednost ponazarja manjšo razdaljo do reference (pri meri GSD pa obratno). Za primerjavo nas zanima le njegova absolutna vrednost.



Slika 6.5: Leva grafa prikazujeta natančnost v odvisnosti od ocene kvalitete za meri PPA (zgoraj) in GSD (spodaj). Rdeča premica predstavlja lienarni model, ki se podatkom najbolje prilega (minimizira vsoto kvadratov napake). Grafa sta ustvarjena na podlagi desnih modelov.

	Fontana	Nagrobnik	Hidrant	Kip
PPA	0,265	0,321	0,407	0,336
GSD	0,144	0,240	0,246	0,218

Tabela 6.1: Absolutne vrednosti koeficientov korelacije med oceno kvalitete in razdaljo do referenčnega modela. Posamezna vrednost predstavlja povprečje koeficientov za vmesne modele med rekonstrukcijo.

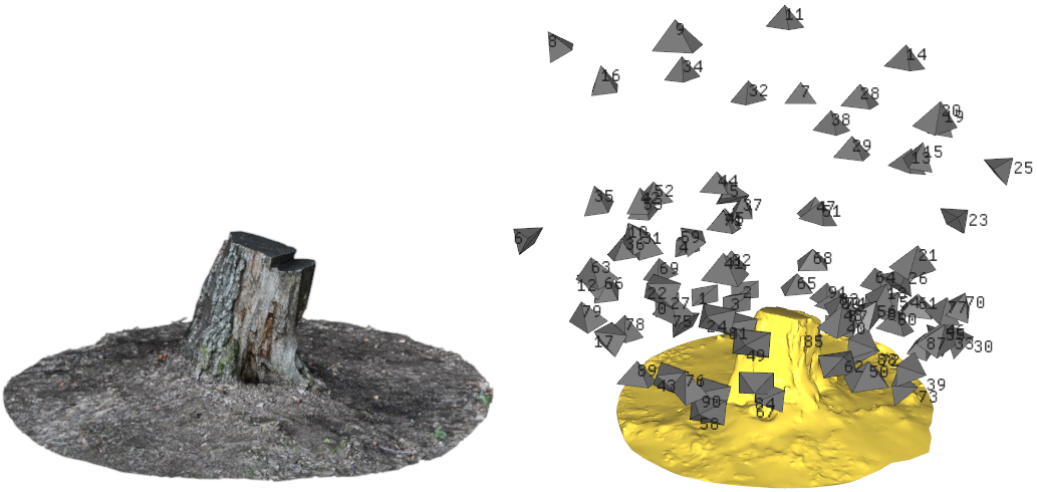
Za vsak testni model smo med rekonstrukcijo izračunali 69 koeficientov korelacije (prvi trije pogledi so uporabljeni za inicializacijo) in na koncu njihove vrednosti povprečili. Rezultati so zbrani v tabeli 6.1. Prikazane so absolutne vrednosti. Koeficient korelacije je za mero PPA v povprečju 1,6-krat večji od mere GSD. Čeprav dejanske vrednosti predstavljajo nizko do srednjo linearno odvisnost spremenljivk, so rezultati nove mere občutno boljši od obstoječe. Pri interpretaciji rezultatov moramo še upoštevati, da odvisnost spremenljivk v resnici ni povsem linearna. Na neki točki namreč novi pogledi nimajo pomembnega prispevka k izboljšavi natančnosti in jo lahko v določenih primerih celo poslabšajo. Naj še omenimo, da je v našem primeru ocena kvalitete izračunana zgolj na podlagi vidnosti trikotnikov. Z vpeljavo dodatnih informacij (npr. lokalni izgled teksture, medsebojna postavitev kamer itd.), je tu še nekaj prostora za izboljšave.

6.4 Evalvacija načrtovanja pogledov

V tem poglavju predstavimo rezultate evalvacije načrtovanja naslednjega pogleda. Gradnjo modela z načrtovanjem pogledov (v nadaljevanju rekonstrukcija NBV) primerjamo z osnovno rekonstrukcijo pri različnih vrednostih parametra ρ. Zanima nas predvsem kako se med gradnjo spreminjata nenatančnost in pokritost modela. Za primerjavo zgradimo 3 osnovne rekonstrukcije modelov z gostotami postavitve $\rho = \{20, 40, 60\}$, ki predstavljajo redko, srednje gosto in gosto enakomerno postavitev kamer. V nekaterih primerih je vrednost $\rho = 20$ premajhna za uspešno rekonstrukcijo, zato jo v takšnih

primerih povečamo na $\rho = 25$. Pri vzorčenju referenčnih modelov smo ponovno uporabili 100000 točk, za rekonstruirane modele pa 2-kratnik števila vozlišč. Za inicializacijo rekonstrukcije NBV smo uporabili prve tri poglede enakomerne postavitve, vsi naslednji pogledi pa so bili izbrani z našim algoritmom za načrtovanje naslednjega pogleda. Parametrov algoritma med evalvacijo nismo spreminjali, njihove vrednosti pa smo določili empirično (poglavje 4.2). Algoritem vrne $n = 20$ predlaganih pogledov sortiranih glede na vrednost cenovne funkcije f_{NBV}. V primeru, da najbolje ocenjeni pogled ni uspešno dodan (npr. neuspešna lokalizacija), se skuša dodati pogled z drugo najboljšo oceno itd. Pri rezultatih poročamo tudi povprečni indeks izbranega naslednjega pogleda, kjer ima prvi predlagani pogled indeks 0. V primeru da nobeden izmed predlaganih pogledov ni uspešno dodan, se rekonstrukcija ustavi (do takšne situacije med evalvacijo ni prišlo). Delovanje smo preizkusili na referenčnih modelih, ki so predstavljeni v poglavju 6.2.

Za začetek si poglejmo tipičen primer pri katerem se rezultati dobro ujemajo s pričakovanji. Na sliki 6.6 je prikazan referenčni model in postavitev kamer, ki je rezultat rekonstrukcije NBV.



Slika 6.6: Slika referenčnega modela štora (leva stran) ter rekonstrukcija in postavitev kamer z načrtovanjem naslednjih pogledov (desna stran).

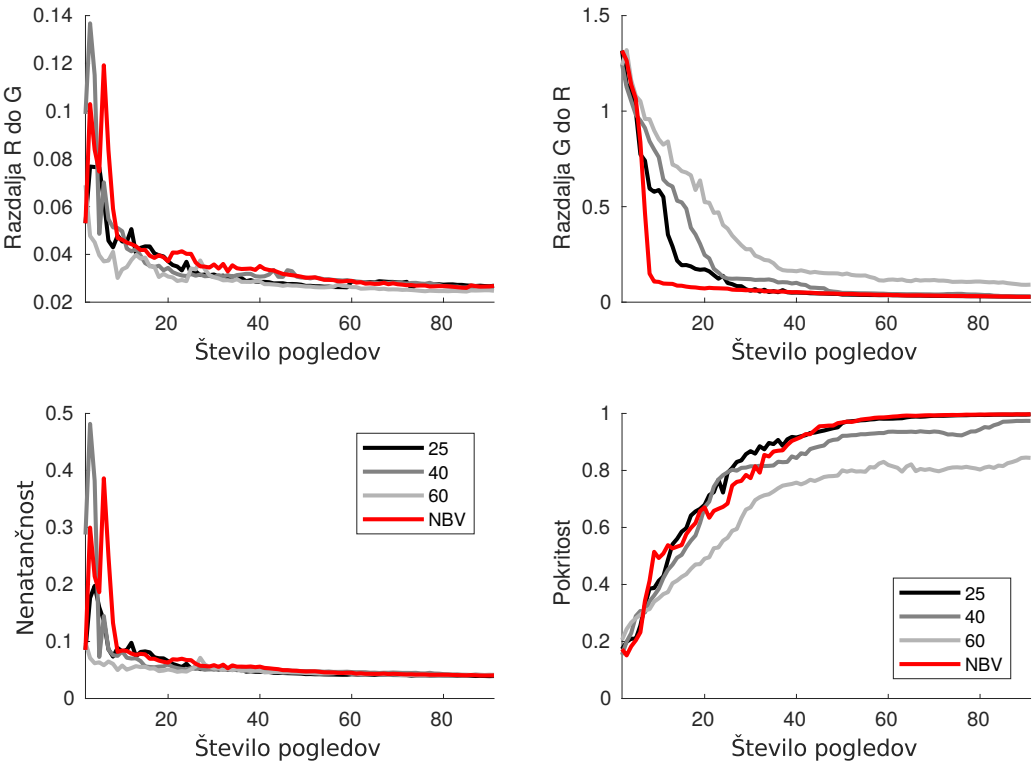
Za ta primer smo na sliki 6.7 prikazali še nekaj primerov slik, ki so uporabljene pri rekonstrukciji, na sliki 6.8 pa slike goste rekonstrukcije upodobljene z istimi legami kamere. Rekonstrukcija je v tem primeru zelo uspešna, zato so razlike med slikami praktično neopazne.

Slika 6.7: Nekaj primerov slik referenčnega modela štora, ki so upodobljene z virtualnimi kamerami in uporabljene pri rekonstrukciji NBV.

Slika 6.8: Primeri slik rekonstruiranega modela.

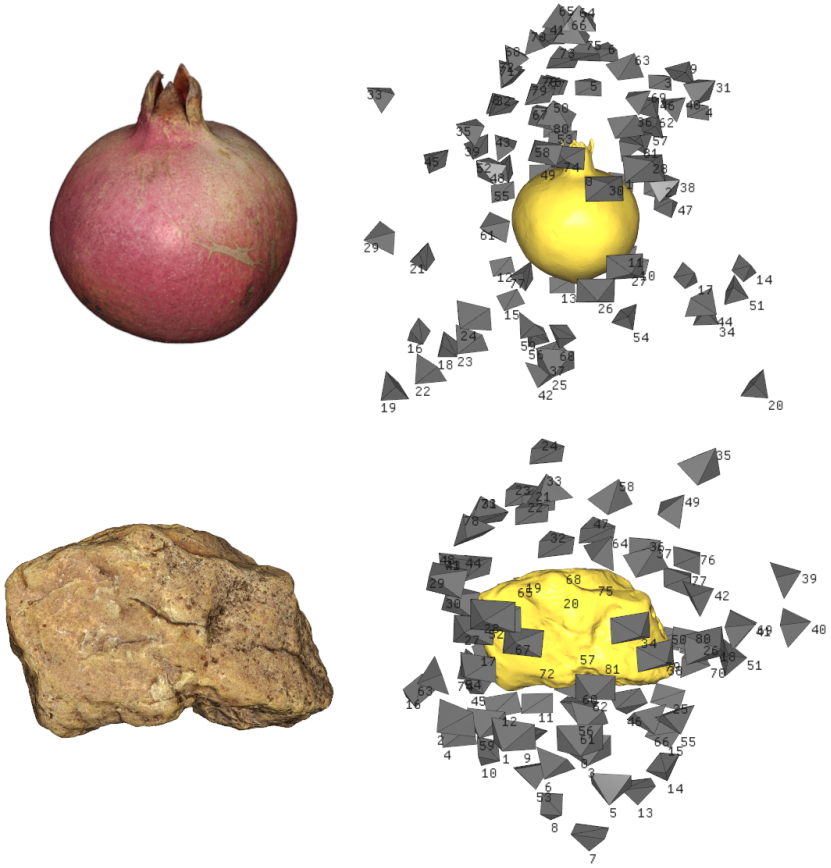
Rezultati primerjave z osnovno rekonstrukcijo so prikazani na sliki 6.9. Za gostoto enakomerne postavitve kamer smo uporabili 3 različne vrednosti. Najmanjša gostota za katero je rekonstrukcija bila uspešna, je v tem primeru 25, poleg tega pa smo uporabili še gostoti 40 in 60. Vključili smo tudi oba grafa razdalj (od R do G in obratno), na podlagi katerih sta izračunani meri

kvalitete. Ker so vse pomembne informacije zajete že v grafih nenatančnosti in pokritosti, bomo v nadaljevanju zgornja grafa izpustili. V tem primeru gre za preprost model, za katerega je enakomerna postavitev zelo učinkovita, kar je razvidno tudi iz grafov. Nenatančnost rekonstrukcije v vseh primerih hitro pade in na koncu doseže skoraj enake vrednosti. Opazimo lahko, da gostota kamer vpliva na hitrost padanja razdalje G do R in posledično na hitrost naraščanja pokritosti, kar je v skladu s pričakovanji. Pri gostejši postavitvi moramo namreč dodati več pogledov, da pokrijemo model. Rezultat rekonstrukcije NBV je dober, saj je primerljiv z osnovno, ki je v tem primeru zelo učinkovita. Dosežena pokritost je skoraj 100%. Povprečni indeks izbranega naslednjega pogleda je 0,06, kar pomeni da je skoraj vedno bil dodan prvi predlagan pogled. Le v štirih primerih je zaradi neuspešne lokalizacije bil izbran naslednji najboljši pogled.



Slika 6.9: Evalvacija rekonstrukcije modela štor.

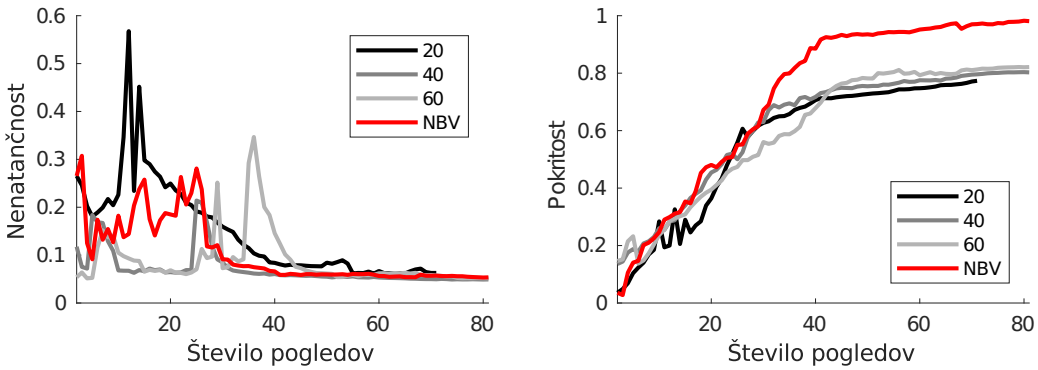
Naslednja testna modela in pripadajoči rekonstrukciji NBV sta prikazana na sliki 6.10. S tema primeroma želimo prikazati večjo fleksibilnost uporabe načrtovanja pogledov v primerjavi s fiksno postavitvijo. Čeprav imata modela preprosto in večinoma konveksno obliko, enakomerna postavitev ne more doseči polne pokritosti, saj spodnji del modelov na kamerah ni viden v celoti.



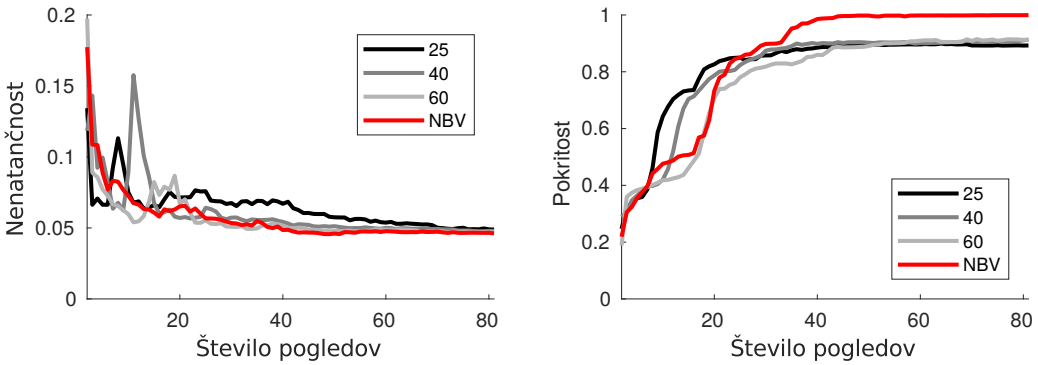
Slika 6.10: Slika referenčnega modela granatnega jabolka (zgoraj) in kamna (spodaj) ter pripadajoči rekonstrukciji NBV (desno).

Rezultati evalvacije za granatno jabolko so prikazani na sliki 6.11. Nenatančnost po 50-tem dodanem pogledu je primerljiva z osnovnimi rekonstrukcijami vseh treh gostot, bolj zanimiva pa je pokritost. Do 30-tega pogleda narašča le za odtenek hitreje od osnovnih, nato pa se povzpne na skoraj polno

pokritost, medtem ko osnovne rekonstrukcije komaj presežejo 80%. Rezultati evalvacije za kamen so prikazani na sliki 6.12, kjer opazimo podobno situacijo. Pokritost sicer v tem primeru na intervalu med 5-im in 20-im pogledom narašča nekoliko počasneje (primerljivo z enakomerno postavitvijo gostote 60), kasneje pa se približa polni pokritosti. Nenatančnost je primerljiva z osnovno rekonstrukcijo. V obeh primerih je razlog za večjo končno pokritost to, da so kamere postavljene tudi na spodnji strani modela kot je to razvidno na sliki 6.10. Povprečni indeks izbranega naslednjega pogleda je pri granatnem jabolku 0,08, pri kamnu pa 0,15, kar pomeni da so ponovno v veliki večini bili uspešno dodani prvi predlagani pogledi.

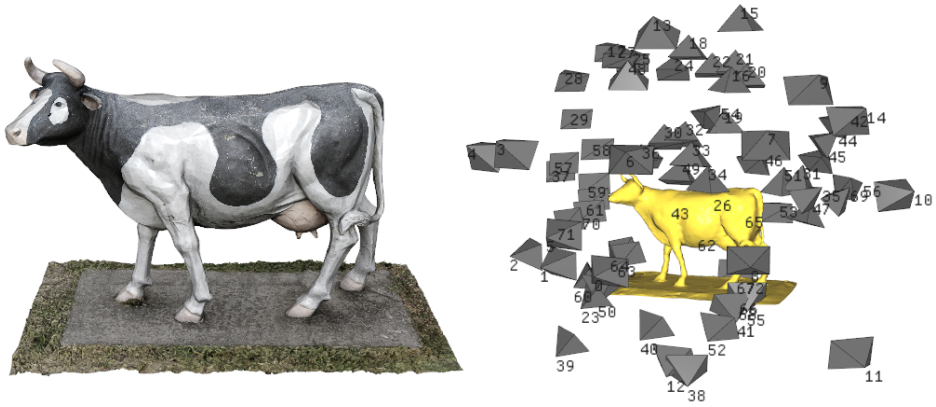


Slika 6.11: Evalvacija rekonstrukcije modela granatno jabolko.



Slika 6.12: Evalvacija rekonstrukcije modela kamen.

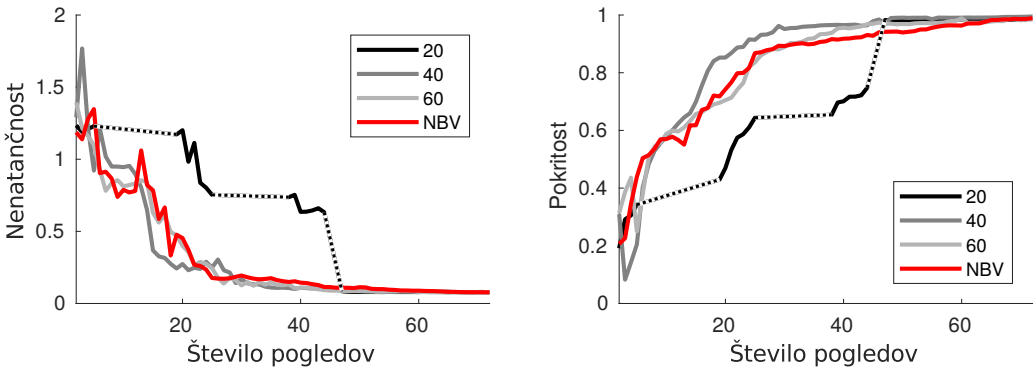
Naslednji primer, s katerim želimo pokazati še eno slabost fiksne postavitve kamer, prikazuje slika 6.13. Zaradi sorazmerno majhne širine modela, se lahko pri redki enakomerni postavitvi zgodi, da kamere niso uspešno dodane k rekonstrukciji.



Slika 6.13: Slika referenčnega modela krave (leva stran) in rekonstrukcija NBV (desna stran).

Omenjena težava je razvidna na sliki 6.14, ki prikazuje rezultate evalvacije. Pri osnovni rekonstrukciji gostote 20 so prvi trije pogledi uspešno dodani. Za naslednji pogled ni najdenih dovolj ujemanj med značilnicami, zato ni uspešno dodan. Enako velja za preostale poglede prvega obroča. Pogledi kljub temu niso odstranjeni, in so dodani, ko je za njih najdenih dovolj ujemanj med značilnicami. V tem primeru se to zgodi, ko se zaključi prvi obroč kamer (oz. pri 20-tem pogledu), kjer se rekonstrukcija lahko nadaljuje. Čeprav do iste težave med rekonstrukcijo pride še dvakrat, so v tem primeru na koncu uspešno dodani vsi pogledi. Če gostoto postavitve povečamo na 40, potem do teh težav ne pride. Nenatančnost rekonstrukcije NBV pada približno enako hitro kot pri osnovni in doseže podobno končno vrednost. Tudi pokritost je v vseh primerih na koncu okoli 99%, med rekonstrukcijo pa je pri uporabi načrtovanja pogledov nekoliko nižja od enakomerne postavitve z gostoto 40. Povprečni indeks izbranega naslednjega pogleda je v tem primeru $3,4$. Razlog za ta precej slabši rezultat pri izbiri pogledov je kompleksnost

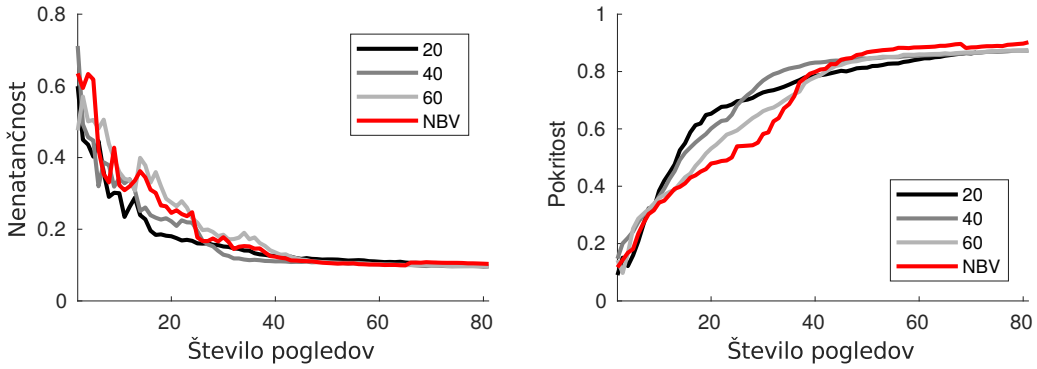
modela. Nekateri predlagani pogledi, ki skušajo razširiti rekonstrukcijo, namreč niso uspešno dodani (zaradi neuspešne lokalizacije ali pa premajhnega števila ujemanj), zato morajo biti uporabljeni manj optimalni pogledi.



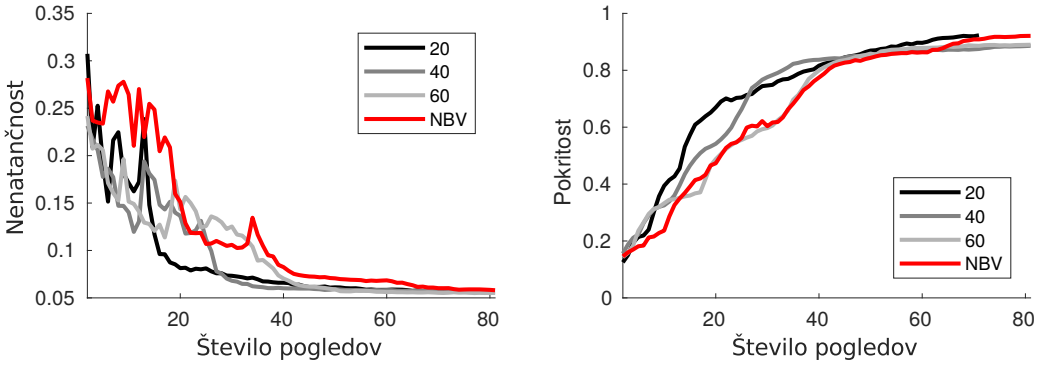
Slika 6.14: Evalvacija rekonstrukcije modela krava.

Naslednji trije modeli so bili izbrani zaradi njihove nekoliko bolj kompleksne geometrije. Prikazani so na sliki 6.15. Z njimi želimo prikazati robustnost naše metode za načrtovanje naslednjega pogleda. Kljub kompleksnejšim modelom, je enakomerna postavitev v teh primerih dokaj ugodna, saj so vsi deli modelov vidni na vsaj eni izmed kamer, poleg tega pa so vsi pogledi uspešno dodani tudi pri redki postavitvi kamer. Grafi nenatančnosti in pokritosti za modele fontane, nagrobnika in kipa, so prikazani na slikah 6.16, 6.17 in 6.18 (v tem vrstnem redu). Rezultati evalvacije so na teh treh primerih podobni. Nekoliko izstopa model kipa, pri katerem je končna nenatančnost boljša od osnovne rekonstrukcije, kljub temu, da je za prvih 40 pogledov slabša. Enako situacijo opazimo tudi pri pokritosti, ki se v tem primeru približa 99%. Vsem trem primerom je skupno to, da pokritost med 10-im in 40-im pogledom nekoliko zaostaja za redkejšo enakomerno postavitvijo in je primerljiva z gosto postavitvijo, končna pokritost pa je enaka ali boljša. Povprečni indeks izbranega naslednjega pogleda je 1,87 pri fontani, 0,72 pri nagrobniku in 1,48 pri kipu. Vrednosti so zaradi kompleksnejših modelov nekoliko večje kot v preprostejših primerih.

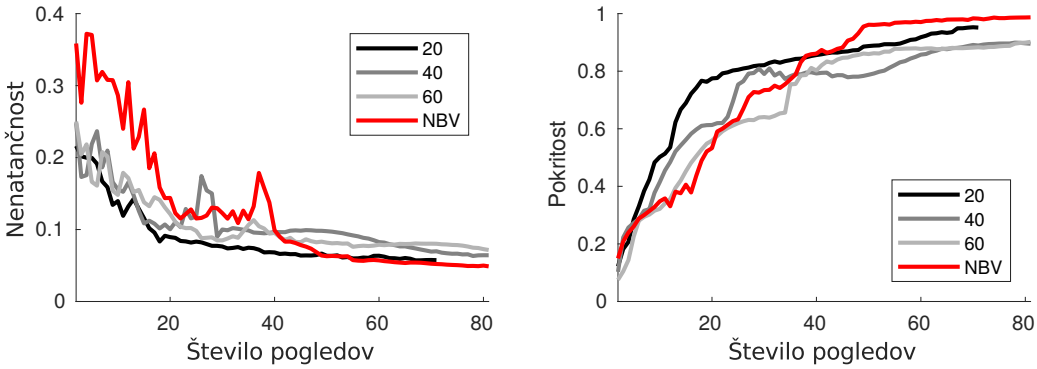
Slika 6.15: Slike referenčnih modelov od zgoraj navzdol: fontana, nagrobnik in kip. Na desni strani so prikazane pripadajoče rekonstrukcije NBV.



Slika 6.16: Evalvacija rekonstrukcije modela fontana.

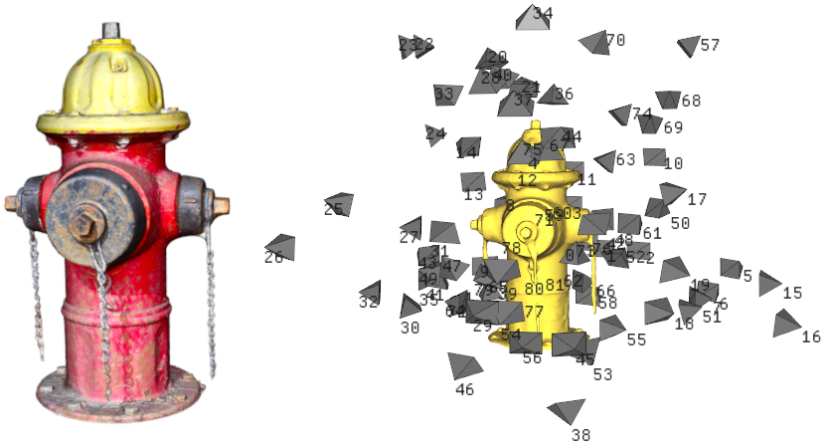


Slika 6.17: Evalvacija rekonstrukcije modela nagrobnik.

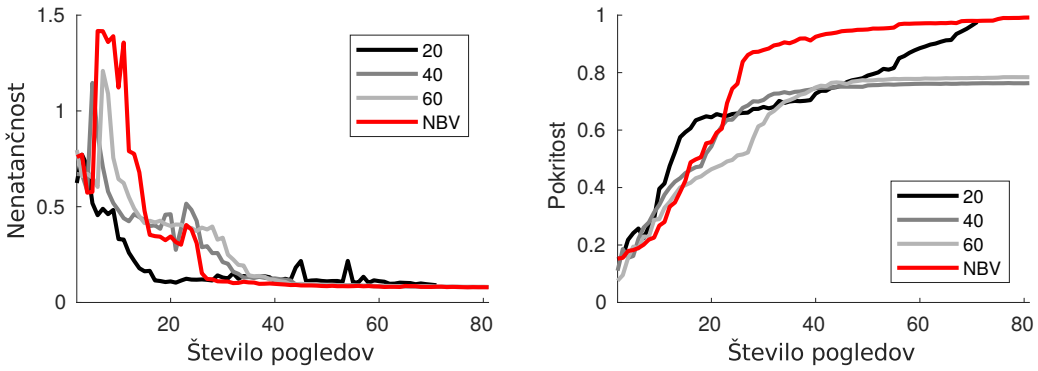


Slika 6.18: Evalvacija rekonstrukcije modela kip.

Naslednji testni model je prikazan na sliki 6.19. Kljub sorazmerno kompleksni geometriji modela je rekonstrukcija NBV nekoliko bolj učinkovita od osnovne. Rezultati evalvacije so prikazani na sliki 6.20. Nenatančnost je na začetku sicer slabša, vendar okoli 30-tega pogleda vrednost pade in postane primerljiva z osnovno rekonstrukcijo. Pokritost je med rekonstrukcijo od 20-tega pogleda dalje precej boljša in na koncu preseže 99%. Tudi povprečni indeks izbranega naslednjega pogleda je v tem primeru dober in znaša 0,1.

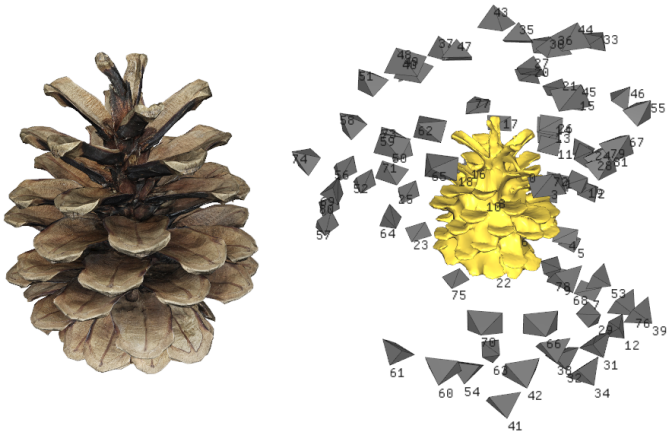


Slika 6.19: Slika referenčnega modela hidranta (leva stran) in rekonstrukcija NBV (desna stran).

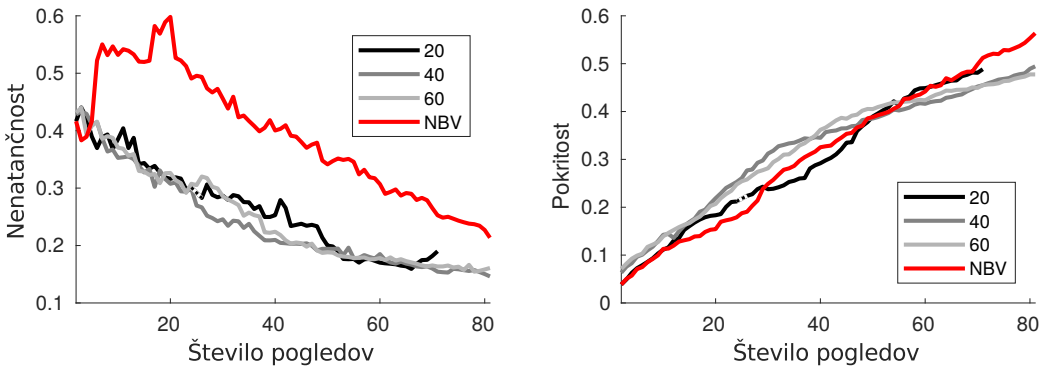


Slika 6.20: Evalvacija rekonstrukcije modela hidrant.

Za konec si poglejmo še primer, ki je za našo metodo načrtovanja pogledov zelo neugoden. Gre za kompleksen model s tanko geometrijo in majhnimi ravnimi površinami. Prikazan je na sliki 6.21, rezultati evalvacije pa na sliki 6.22. Kljub temu, da je pokritost primerljiva z osnovno, le ta narašča počasi in pri 80-tem pogledu ne preseže 60%. Nenatančnost je med rekonstrukcijo ves čas precej večja od osnovne. Tudi povprečni indeks izbranega naslednjega pogleda je visok in znaša 3,48. Razlog za to je ponovno velika kompleksnost modela in posledično neuspešna lokalizacija nekaterih kamer.



Slika 6.21: Slika referenčnega modela storža (leva stran) in rekonstrukcija NBV (desna stran).

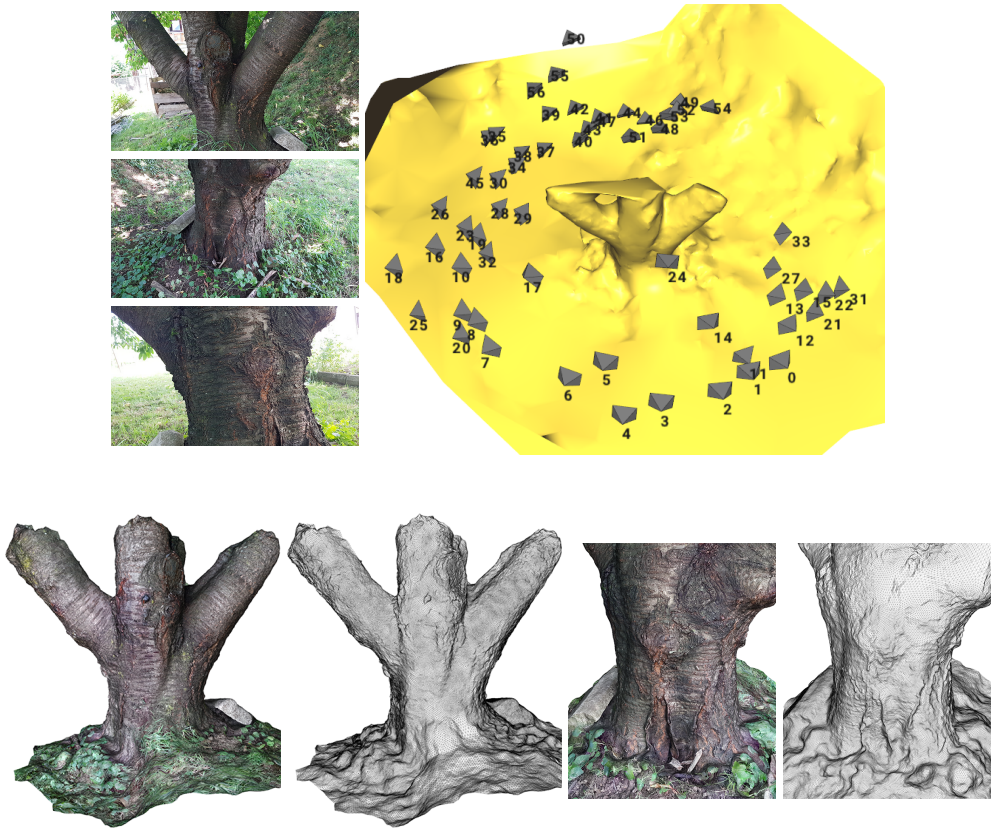


Slika 6.22: Evalvacija rekonstrukcije modela storž.

6.5 Primeri rekonstrukcij resničnih predmetov

V tem poglavju predstavimo še nekaj rekonstrukcij resničnih predmetov in časovnih meritev. Za inicializacijo rekonstrukcije smo po lastni presoji izbrali nekaj pogledov in določili območje interesa. Pri dodajanju novih slik smo upoštevali predlagane poglede. Lokalizacija predlaganega pogleda je včasih neuspešna, ali pa natančna postavitev kamere zaradi fizičnih omejitev prostora ni mogoča. V takšnih primerih smo izbrali naslednji predlagani pogled, v rezultatih pa poročamo o povprečnem indeksu izbranega pogleda. Poleg tega poročamo še o absolutni in relativni napaki, ki jo naredimo pri postavitvi kamere. Absolutna napaka pozicije kamere je izračunana na podlagi meritev resničnega predmeta, relativno napako pa izračunamo kot delež razdalje dveh najbolj oddaljenih kamer. Napako orientacije predstavlja kot med optičnima osema predlagane in resnične postavitve kamere. Ločljivost zajetih slik je fiksna in znaša 2560×1440, kamera pa je predhodno kalibrirana. Ker referenčni modeli tokrat niso na voljo, lahko kvaliteto rekonstrukcije ocenimo zgolj vizualno.

Prvi primer prikazuje rekonstrukcijo debla drevesa, za katero smo uporabili 57 slik. Na zgornjem delu slike 6.23 je prikazana postavitev kamer in nekaj primerov zajetih slik. Povprečni indeks naslednjega izbranega pogleda znaša 0,8, relativna napaka pozicije 6,5%, absolutna napaka pozicije 20 cm in napaka orientacije 13,6 °. Končni 3D model, ki je rezultat goste rekonstrukcije po odstranitvi odvečnih trikotnikov, je prikazan na spodnjem delu slike 6.23. Geometrija debla se vizualno dobro ujema z njegovim resničnim videzom. Rekonstrukcija travnate površine in ostalih rastlin v okolici sicer ni naš cilj, opazimo pa lahko, da so ta področja bolj problematična in manj natančno rekonstruirana.



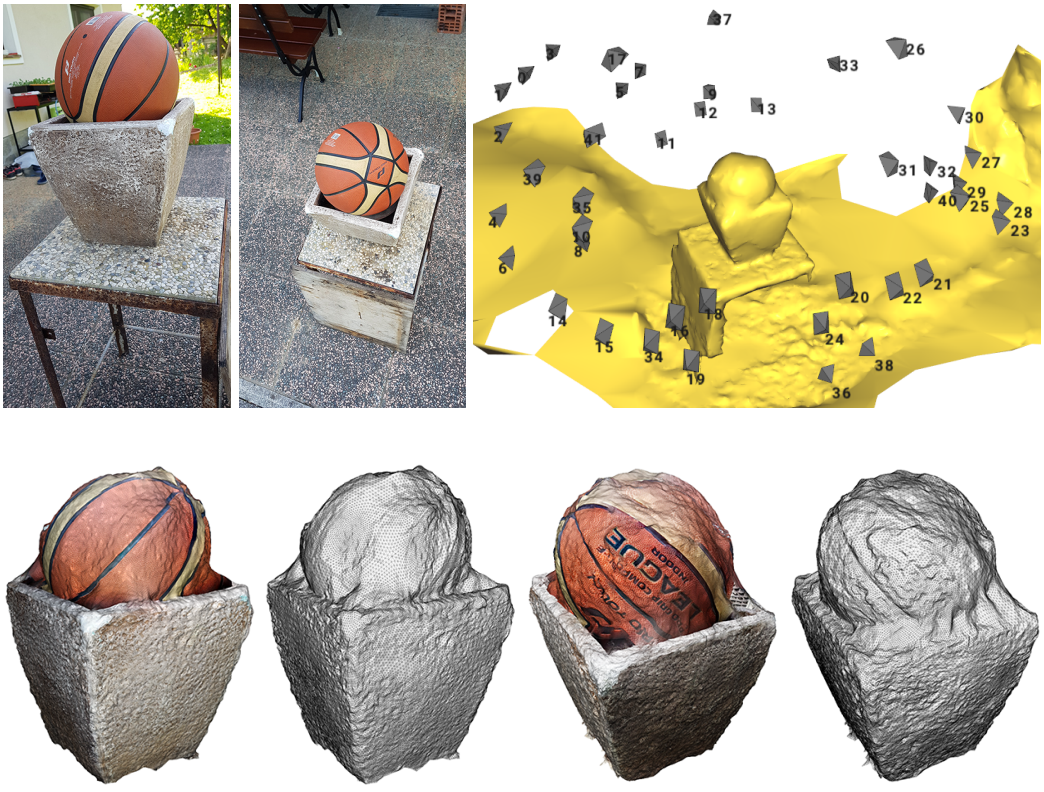
Slika 6.23: Nekaj primerov slik uporabljenih pri rekonstrukciji (levo zgoraj), postavitev kamer (desno zgoraj) in rezultat goste rekonstrukcije debla (spodaj).

Za naslednji primer smo izbrali nekoliko manjši predmet in sicer gorske čevlje. Rezultat redke rekonstrukcije je prikazan na zgornjem delu slike 6.24. Pri zajemanju slik je bila 4-krat rekonstrukcija površine neuspešna. Ker je načrtovanje pogledov v takšnem primeru nemogoče, smo 4 slike dodali po lastni presoji. Skupno smo zajeli 31 slik. Povprečni indeks naslednjega izbranega pogleda znaša 0,48, relativna napaka pozicije 12,8%, absolutna napaka pozicije 18 cm in napaka orientacije 15,3 °. Končni 3D model je prikazan na spodnjem delu slike 6.24. Na slabše teksturiranih predelih čevlja sicer lahko opazimo nekaj nepravilnosti, v splošnem pa se geometrija vizualno dobro ujema z resničnim predmetom.

Slika 6.24: Nekaj primerov slik uporabljenih pri rekonstrukciji (levo zgoraj), postavitev kamer (desno zgoraj) in rezultat goste rekonstrukcije gorskih čevljev (spodaj).

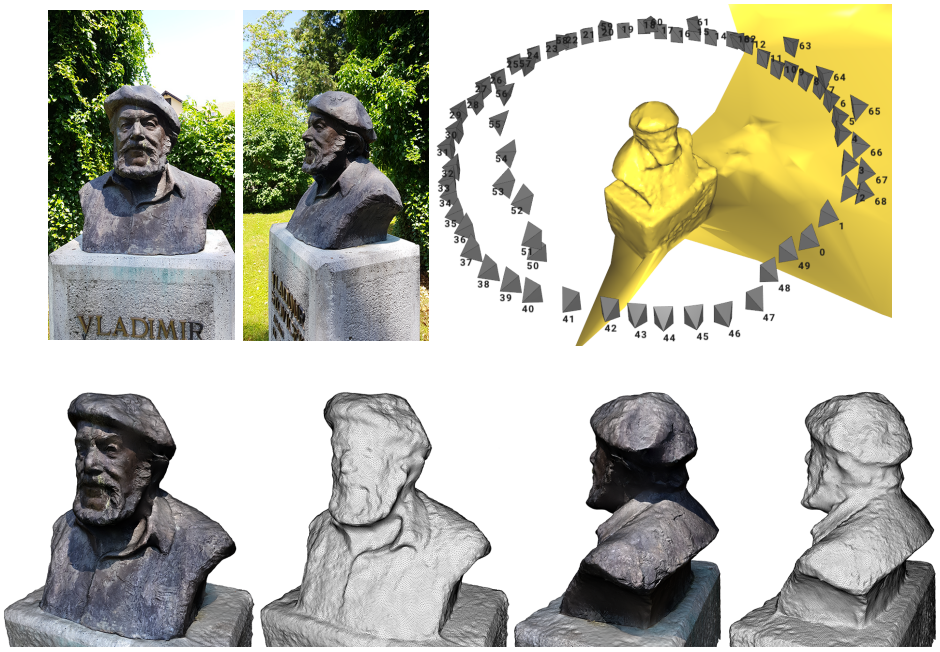
Redka rekonstrukcija naslednjega primera je prikazana na zgornjem delu slike 6.25. Čeprav je oblika predmeta navidez preprosta, je zaradi ostrih robov in nekoliko manj ugodne teksture žoge, ta primer bolj težaven za re-

konstrukcijo z načrtovanjem naslednjih pogledov. Zajeli smo 43 slik, od tega smo 3 slike morali dodati po lastni presoji, saj nobeden od predlaganih pogledov ni bil uspešno lokaliziran. Povprečni indeks naslednjega izbranega pogleda znaša 1,1, relativna napaka pozicije 7,7%, absolutna napaka pozicije 15 cm in napaka orientacije 15,7°. Končni 3D model je prikazan na spodnjem delu slike 6.25. Rekonstrukcija spodnjega dela (lonec) je zelo uspešna in se vizualno dobro ujema z resničnim predmetom. Nekoliko več nepravilnosti lahko opazimo na žogi. Rekonstruirana površina je pregroba, oblika pa ni povsem sferična, kar je najbolj razvidno na zgornjem delu. Razlog za te nepravilnosti je nekoliko manj ugodna tekstura.



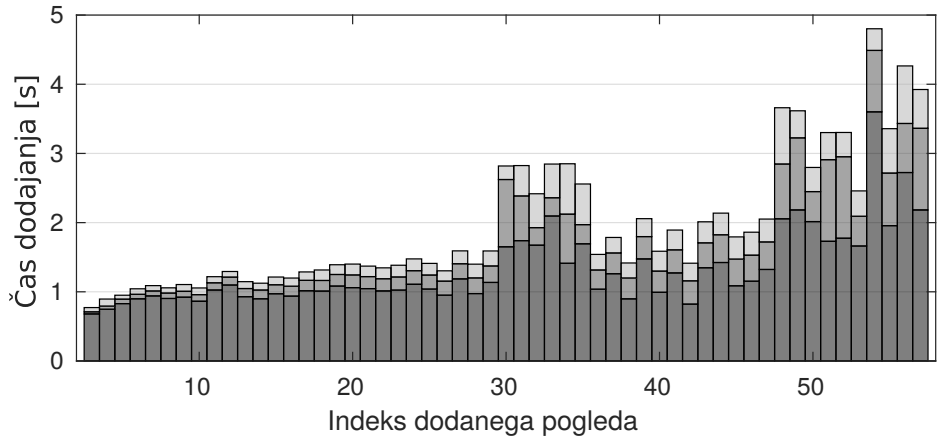
Slika 6.25: Nekaj primerov slik uporabljenih pri rekonstrukciji (levo zgoraj), postavitev kamer (desno zgoraj) ter rezultat goste rekonstrukcije lonca in žoge (spodaj).

Zadnji primer rekonstrukcije smo zgradili brez načrtovanja pogledov. Slike smo zajeli po lastni presoji in jih v našo programsko rešitev naložili z diska. Rezultat redke rekonstrukcije je prikazan na zgornjem delu slike 6.26, končni 3D model pa na spodnjem delu. Zaradi zelo ugodne teksture predmeta in dovolj velikega števila zajetih slik (68), se geometrija dobro ujema resničnim predmetom.



Slika 6.26: Nekaj primerov slik uporabljenih pri rekonstrukciji (levo zgoraj), postavitev kamer (desno zgoraj) in rezultat goste rekonstrukcije kipa (spodaj).

Za konec si poglejmo še časovne meritve za prvi primer rekonstrukcije v tem podpoglavju (deblo). Delovanje smo testirali na prenosnem računalniku s procesorjem Intel® Core™ i5-7300HQ (4 jedra), 16GB glavnega pomnilnika in grafično kartico NVIDIA® GeForce® GTX 1050 (2GB). Meritve prikazuje graf na sliki 6.27. Sestavljen je iz treh slojev. Prvi sloj predstavlja čas dodajanja nove slike k rekonstrukciji, naslednji je čas rekonstrukcije površine, zadnji pa je čas, ki je potreben za načrtovanje naslednjega pogleda.



Slika 6.27: Časovna meritev rekonstrukcije. Sloji grafa predstavljajo čas dodajanja nove slike, rekonstrukcije površine in načrtovanja pogledov.

Celoten čas procesiranja nove slike sicer narašča linearno, vendar dovolj počasi, da je takšen postopek primeren za rekonstrukcijo s sprotnim odzivom, za predmete manjše velikosti. V tem primeru čas razširitve rekonstrukcije ne preseže 5 s. Naj še omenimo, da je čas dodajanja pogleda odvisen od njegovega prispevka pri rekonstrukciji. V primeru da pogled prispeva veliko novih točk, je čas dodajanja sorazmerno daljši kot pa v primeru kjer pogled nima pomembnega prispevka. Ta pojav lahko opazimo npr. pri dodajanju 54-tega pogleda, kjer je čas procesiranja nekoliko daljši od pričakovanega. Gosta rekonstrukcija je časovno bolj zahtevna. V tem primeru je čas izboljšave ločljivosti modela 1101 s, čas teksturiranja pa 338 s.

Poglavje 7

Sklepne ugotovitve

V tem delu smo z uporabo odprtokodnih knjižnic in sodobnih algoritmov implementirali celoten postopek rekonstrukcije, ki iz vhodnih barvnih slik proizvede natančen in teksturiran 3D model. Algoritmi rekonstrukcije temeljijo na določenih predpostavkah, zato so vhodne slike kritičnega pomena pri doseganju želene kvalitete 3D modela. Razvita programska oprema preko uporabniškega vmesnika omogoča interaktivno izkušnjo in uporabniku nudi informacije o trenutnem stanju rekonstrukcije. S temi informacijami lahko uporabnik optimizira proces zajemanja slik.

Poleg omenjene programske opreme je prispevek našega dela tudi zasnova in implementacija nove metode za načrtovanje najboljšega naslednjega pogleda, ki temelji na lastni meri za oceno kvalitete 3D modela. Za delovanje potrebuje le trenutni 3D model predstavljen s trikotniško mrežo in informacije o legi predhodno dodanih kamer, zato je metoda dovolj splošna in je lahko uporabljena tudi kot gradnik pri avtonomni rekonstrukciji z različnimi roboti (npr. kvadrokopter, robotski manipulator itd.).

Cilje, zadane v tem delu, smo dosegli, kljub temu pa ostaja nekaj prostora za izboljšave. Pomembna izboljšava uporabniške izkušnje, bi bila implementacija sistema po modelu "odjemalec – strežnik". Glavni del rekonstrukcije, ki je časovno bolj zahteven, bi tako potekal na strežniku, aplikacija odjemalca pa bi se izvajala neposredno na mobilnem telefonu in bi omogočala prikaz 3D

modela in trenutne lege kamere v realnem času ter zajemanje novih slik. Naslednja ideja za izboljšavo je ocena kvalitete 3D modela s pomočjo strojnega učenja. Simulirano okolje, ki smo ga razvili za evalvacijo, omogoča rekonstrukcijo virtualnih predmetov in izračun dejanske natančnosti rekonstruiranega 3D modela po vsaki dodani sliki. Na ta način bi lahko generirali veliko količino podatkov in jih uporabili za nadzorovano učenje umetne nevronske mreže. Tudi pri načrtovanju naslednjih pogledov bi v cenovno funkcijo bilo mogoče vpeljati dodatne informacije o videzu teksture in vidnosti rekonstruiranih značilnic. S temi dodatnimi informacijami bi verjetno lahko zmanjšali število predlaganih pogledov, ki so pri rekonstrukciji neuspešno lokalizirani.

Kljub možnim izboljšavam, so rezultati evalvacije pokazali uspešno delovanje našega sistema. Naša mera za oceno kvalitete ima na testiranih modelih boljši koeficient linearne korelacije z dejansko natančnostjo od obstoječe mere. Pokažemo tudi, da je kvaliteta rekonstrukcije z načrtovanjem naslednjega pogleda primerljiva in v nekaterih primerih boljša od rekonstrukcije z enakomerno postavljenimi kamerami po navidezni polkrogli. Pokritost sicer večinoma narašča nekoliko počasneje, končna vrednost pa je primerljiva ali višja. Za razliko od fiksne postavitve kamer, naša metoda nima nobenih posebnih predpostavk o obliki in velikosti predmeta. Uspešno delovanje rekonstrukcije smo prikazali tudi na resničnih predmetih. Primeren pristop, ki ga v našem delu sicer nismo testirali, je rekonstrukcija na kombiniran način. Tako bi na začetku zajeli nekaj slik z enakomerno postavitvijo kamere, ko pa bi dosegli dobro pokritost, bi pri dodajanju upoštevali predlagane poglede in s tem izboljšali natančnost rekonstrukcije.

Literatura

[1] S. M. Seitz, B. Curless, J. Diebel, D. Scharstein, R. Szeliski, A comparison and evaluation of multi-view stereo reconstruction algorithms, in: null, IEEE, 2006, pp. 519–528.

[2] C. Hoppe, M. Klopschitz, M. Rumpler, A. Wendel, S. Kluckner, H. Bischof, G. Reitmayr, Online feedback for structure-from-motion image acquisition., in: BMVC, Vol. 2, 2012, p. 6.

[3] M. Pollefeys, L. Van Gool, M. Vergauwen, F. Verbiest, K. Cornelis, J. Tops, R. Koch, Visual modeling with a hand-held camera, International Journal of Computer Vision 59 (3) (2004) 207–232.

[4] N. Snavely, S. M. Seitz, R. Szeliski, Photo tourism: exploring photo collections in 3d, in: ACM transactions on graphics (TOG), Vol. 25, ACM, 2006, pp. 835–846.

[5] C. Wu, Towards linear-time incremental structure from motion, in: 3D Vision-3DV 2013, 2013 International Conference on, IEEE, 2013, pp. 127–134.

[6] S. Agarwal, N. Snavely, I. Simon, S. M. Seitz, R. Szeliski, Building rome in a day, in: 2009 IEEE 12th international conference on computer vision, IEEE, 2009, pp. 72–79.

[7] Y. Furukawa, C. Hernández, et al., Multi-view stereo: A tutorial, Foundations and Trends® in Computer Graphics and Vision 9 (1-2) (2015) 1–148.

[8] R. Mur-Artal, J. D. Tardós, Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras, IEEE Transactions on Robotics 33 (5) (2017) 1255–1262.

[9] R. A. Newcombe, S. J. Lovegrove, A. J. Davison, Dtam: Dense tracking and mapping in real-time, in: 2011 international conference on computer vision, IEEE, 2011, pp. 2320–2327.

[10] S. Daftry, C. Hoppe, H. Bischof, Building with drones: Accurate 3d facade reconstruction using mavs, in: 2015 IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2015, pp. 3487–3494.

[11] S. Zollmann, C. Hoppe, T. Langlotz, G. Reitmayr, Flyar: Augmented reality supported micro aerial vehicle navigation, IEEE transactions on visualization and computer graphics 20 (4) (2014) 560–568.

[12] S. Kriegel, C. Rink, T. Bodenmüller, M. Suppa, Efficient next-best-scan planning for autonomous 3d surface reconstruction of unknown objects, Journal of Real-Time Image Processing 10 (4) (2015) 611–631.

[13] R. Pito, A solution to the next best view problem for automated surface acquisition, IEEE Transactions on Pattern Analysis and Machine Intelligence 21 (10) (1999) 1016–1030.

[14] M. Krainin, B. Curless, D. Fox, Autonomous generation of complete 3d object models using next best view manipulation planning, in: Robotics and Automation (ICRA), 2011 IEEE International Conference on, IEEE, 2011, pp. 5031–5037.

[15] E. Dunn, J.-M. Frahm, Next best view planning for active model improvement., in: BMVC, 2009, pp. 1–11.

[16] J. A. Nelder, R. Mead, A simplex method for function minimization, The computer journal 7 (4) (1965) 308–313.

[17] C. Wu, Visualsfm: A visual structure from motion system, `http://ccwu.me/vsfm/`.

[18] J. L. Schönberger, J.-M. Frahm, Structure-from-motion revisited, in: Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

[19] Z. Zhang, et al., Flexible camera calibration by viewing a plane from unknown orientations., in: Iccv, Vol. 99, 1999, pp. 666–673.

[20] J. Heikkila, O. Silven, A four-step camera calibration procedure with implicit image correction, in: cvpr, IEEE, 1997, p. 1106.

[21] P. Moulon, P. Monasse, R. Marlet, Global fusion of relative motions for robust, accurate and scalable structure from motion, in: Proceedings of the IEEE International Conference on Computer Vision, 2013, pp. 3248–3255.

[22] D. G. Lowe, Distinctive image features from scale-invariant keypoints, International journal of computer vision 60 (2) (2004) 91–110.

[23] T. Sattler, T. Weyand, B. Leibe, L. Kobbelt, Image retrieval for image-based localization revisited., in: BMVC, Vol. 1, 2012, p. 4.

[24] J. Cheng, C. Leng, J. Wu, H. Cui, H. Lu, Fast and accurate image matching with cascade hashing for 3d reconstruction, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 1–8.

[25] M. A. Fischler, R. C. Bolles, Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography, Communications of the ACM 24 (6) (1981) 381–395.

[26] J. L. Schonberger, H. Hardmeier, T. Sattler, M. Pollefeys, Comparative evaluation of hand-crafted and learned local features, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 1482–1491.

[27] E. Weitz, Sift - scale-invariant feature transform, `http://weitz.de/sift/` (2016).

[28] R. Arandjelović, A. Zisserman, Dislocation: Scalable descriptor distinctiveness for location recognition, in: Asian Conference on Computer Vision, Springer, 2014, pp. 188–204.

[29] H. Jegou, M. Douze, C. Schmid, Hamming embedding and weak geometric consistency for large scale image search, in: European conference on computer vision, Springer, 2008, pp. 304–317.

[30] H. Jégou, M. Douze, C. Schmid, On the burstiness of visual elements, in: 2009 IEEE Conference on Computer Vision and Pattern Recognition, IEEE, 2009, pp. 1169–1176.

[31] H. Stewenius, C. Engels, D. Nistér, Recent developments on direct relative orientation, ISPRS Journal of Photogrammetry and Remote Sensing 60 (4) (2006) 284–294.

[32] M. Byröd, K. Josephson, K. Åström, Fast and stable polynomial equation solving and its application to computer vision, International Journal of Computer Vision 84 (3) (2009) 237–256.

[33] R. Hartley, A. Zisserman, Multiple view geometry in computer vision, Cambridge university press, 2003.

[34] L. Kneip, D. Scaramuzza, R. Siegwart, A novel parametrization of the perspective-three-point problem for a direct computation of absolute camera position and orientation.

[35] R. Szeliski, Computer vision: algorithms and applications, Springer Science & Business Media, 2010.

[36] R. I. Hartley, P. Sturm, Triangulation, Computer vision and image understanding 68 (2) (1997) 146–157.

[37] B. Triggs, P. F. McLauchlan, R. I. Hartley, A. W. Fitzgibbon, Bundle adjustment—a modern synthesis, in: International workshop on vision algorithms, Springer, 1999, pp. 298–372.

[38] J. J. Moré, The levenberg-marquardt algorithm: implementation and theory, in: Numerical analysis, Springer, 1978, pp. 105–116.

[39] M. Jancosek, T. Pajdla, Exploiting visibility information in surface reconstruction to preserve weakly supported surfaces, International scholarly research notices 2014.

[40] P. Labatut, J.-P. Pons, R. Keriven, Robust and efficient surface reconstruction from range data, in: Computer graphics forum, Vol. 28, Wiley Online Library, 2009, pp. 2275–2290.

[41] M. Kazhdan, M. Bolitho, H. Hoppe, Poisson surface reconstruction, in: Proceedings of the fourth Eurographics symposium on Geometry processing, Vol. 7, 2006.

[42] J.-D. Boissonnat, Geometric structures for three-dimensional shape representation, ACM Transactions on Graphics (TOG) 3 (4) (1984) 266–286.

[43] H.-H. Vu, P. Labatut, J.-P. Pons, R. Keriven, High accuracy and visibility-consistent dense multiview stereo, IEEE transactions on pattern analysis and machine intelligence 34 (5) (2012) 889–901.

[44] J.-P. Pons, R. Keriven, O. Faugeras, Multi-view stereo reconstruction and scene flow estimation with a global image-based matching score, International Journal of Computer Vision 72 (2) (2007) 179–193.

[45] M. Waechter, N. Moehrle, M. Goesele, Let there be color! large-scale texturing of 3d reconstructions, in: European Conference on Computer Vision, Springer, 2014, pp. 836–850.

[46] Y. Boykov, O. Veksler, R. Zabih, Fast approximate energy minimization via graph cuts, in: Proceedings of the Seventh IEEE International Conference on Computer Vision, Vol. 1, IEEE, 1999, pp. 377–384.

[47] C. Sweeney, Theia multiview geometry library: Tutorial & reference, http://theia-sfm.org.

[48] W. Changchang, Siftgpu: a gpu implementation of scale invariant feature transform (sift), https://github.com/pitzer/SiftGPU (2007).

[49] Openmvs: open multi-view stereo reconstruction library, https://github.com/cdcseacave/openMVS.

[50] A. Jacobson, D. Panozzo, et al., libigl: A simple C++ geometry processing library, http://libigl.github.io/libigl/ (2018).

[51] O. Cornut, Dear imgui, https://github.com/ocornut/imgui.

[52] C. Guillemet, Imguizmo, https://github.com/CedricGuillemet/ImGuizmo.

[53] R. Jensen, A. Dahl, G. Vogiatzis, E. Tola, H. Aanæs, Large scale multi-view stereopsis evaluation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 406–413.

[54] P. J. Besl, N. D. McKay, Method for registration of 3-d shapes, in: Sensor Fusion IV: Control Paradigms and Data Structures, Vol. 1611, International Society for Optics and Photonics, 1992, pp. 586–607.

[55] S. Umeyama, Least-squares estimation of transformation parameters between two point patterns, IEEE Transactions on Pattern Analysis & Machine Intelligence (4) (1991) 376–380.

[56] J. L. Blanco, P. K. Rai, nanoflann: a C++ header-only fork of FLANN, a library for nearest neighbor (NN) with kd-trees, https://github.com/jlblancoc/nanoflann (2014).